PROCESS CHOICE DECISIONS IN MANUFACTURING

BY

GEORGE L. VAIRAKTARAKIS

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA
1994

UNIVERSITY OF FLORIDA LIBRARIES

ACKNOWLEDGEMENTS

The author gratefully acknowledges Dr. Chung-Yee Lee, his committee chairman, for all his expert guidance, infinite patience, constant encouragement, and advice at any time throughout this study. Special appreciation is expressed to Dr. P. Kouvelis for his valuable comments and suggestions.

The author would also like to express his appreciation to the Department of Industrial and Systems Engineering at the University of Florida and in particular its chairman Dr. D. Jack Elzinga for sponsoring his studies. Finally, the author wishes to express the most heartfelt thanks to his wife Panagiota Kabouri for her patience, support and continuous encouragement during the course of this study.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS ii

ABSTRACT vii

CHAPTERS 1

1 INTRODUCTION 1

1.1 Motivation Of Study 1

1.2 Operating Environments 5

1.2.1 The Single Machine Model 5

1.2.2 The Two-Stage Hybrid Flowshop Model 6

1.2.3 The Multi-Stage Hybrid Flowshop Model 7

1.2.4 The Two-Stage Flexible Flowshop Model 7

1.2.5 The Two-Stage Flexible Jobshop Model 7

1.2.6 The Assembly Line Model 8

1.3 Outline 8

2 DESIGN FOR SCHEDULABILITY: FLEXIBLE LOOK-BEHIND AND LOOK-AHEAD FLOWSHOPS 12

2.1 Introduction 12

2.2 The Flexible LBFS and LAFS Designs 18

2.3 Heuristic Algorithms for LBFS and LAFS 19

2.3.1 Average Performance of Heuristics 23
2.4 Design for Schedulability 25
2.4.1 Analytical Comparisons 26
2.4.2 Average Comparisons 32
2.5 Comparative Results 41
2.6 Discussion 43
2.7 Conclusion 45

3 MINIMIZING MAKESPAN IN HYBRID FLOWSHOPS 46

3.1 Introduction 46
3.2 Problem Description and Literature Review 48
3.3 Heuristic Algorithms for the Hybrid Flowshop 50
3.3.1 Two Stage Production 50
3.3.2 A Heuristic for the Hybrid Flowshop 58
3.4 Conclusion 59

4 THE TWO-STAGE FLEXIBLE FLOWSHOP ENVIRONMENT 61

4.1 Introduction 61
4.2 The Two Stage Flexible Flowshop Problem 64
4.2.1 Problem Description 64
4.2.2 Properties of Optimal 2FFS Schedules 66
4.3 A Dynamic Programming Algorithm for the 2FFS 68
4.3.1 Description of the DP Algorithm 68
4.3.2 Computational Performance of the DP Algorithm 70
4.4 Heuristic Algorithms for the 2FFS 71
4.4.1 A Modified Parallel Machine Heuristic 71
4.4.2 Performance of Johnson's Algorithm as a Heuristic for the 2FFS 77

4.5 Flexible Flowshops versus Traditional Flowshops and Parallel Machines 79
4.6 Conclusion 86

5 THE TWO-STAGE FLEXIBLE JOBSHOP SCHEDULING PROBLEM . 89

5.1 Introduction 89
5.2 The Two Stage Flexible Job Shop Problem 91
5.2.1 Problem Description 91
5.2.2 Properties of Optimal 2FJS Schedules 94
5.3 A Dynamic Programming Algorithm for the 2FFS 96
5.3.1 Description of the DP Algorithm 96
5.4 Flexible Job Shop versus Traditional Flowshops and Parallel Machines 99

6 MANPOWER PLANNING IN MIXED MODEL TRANSFER LINES . . . 102

6.1 Introduction 102
6.2 Description of the Assembly Line Problem 107
6.3 The 2SAL Problem 109
6.3.1 Formulation of 2SAL 109
6.3.2 Solution Procedure for 2SAL 111
6.4 Complexity Result 117
6.5 Lower Bounds and Heuristics 120
6.6 Computational Results 130
6.7 Extensions 135
6.8 Conclusion 137

7 FUTURE RESEARCH 139

REFERENCES 140

BIOGRAPHICAL SKETCH 146

Abstract of Dissertation Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Doctor of Philosophy

PROCESS CHOICE DECISIONS IN MANUFACTURING

By

George L. Vairaktarakis

August 1994

Chairman: Chung-Yee Lee
Major Department: Industrial and Systems Engineering

Our study sheds light to the intriguing interactions between the various factors that affect the performance of a manufacturing system. Such factors include machine characteristics, workorder specifications, machine layout, routing control, and complexity of production management. The problems treated in this dissertation are motivated from a variety of real world problems arising from *PCB assembly, the glass container industry, cable manufacturing, furniture industry, fire truck and automobile assembly*, etc.

We develop a two step framework for the *evaluation of flexibility.* In the identification step we identify various *modes of flexibility.* We treat three such modes; *dedicated machines*, *processing flexibility* and *routing flexibility.* In the quantification step we quantify the *value of flexibility* of various process designs in terms of manufacturing performance defined as *machine utilization* and *throughput performance.*

We first study the hybrid flowshop with minimum makespan objective. This flowshop environment consists of multiple stages and multiple machines in each stage. A task within a stage can be processed by any of the machines in the stage. Our objective is to schedule a set of n jobs so as to minimize makespan.

Then, we address a scheduling problem for a two stage flexible flowshop (2FFS) environment with significant processing flexibility across stages and one machine per stage. A job can be either completely processed at one of the stages or its processing is split in a prespecified fashion between stages. Machines are assumed to be equipped with sophisticated tool magazines that can perform one or more operations of the final product.

Finally we study an assembly line design. The design considered is called *mixed model transfer line*. Each job spends a predetermined production cycle of c periods in each station and this cycle is the same for all stations. The workforce level required within each station is such that a particular task can be completed within c periods. Our objective is to minimize the maximum number of workers required over any production cycle.

CHAPTER 1

INTRODUCTION

1.1 Motivation Of Study

A little less than 10 years ago, industry shifted towards *design for manufacturing.* This shift led over 50% of US manufacturing to utilize flexible machines that are able to produce small to medium batches of a variety of similar or related products. This way the industry could readily respond to a more demanding customer cultivated by the numerous successes of manufacturing in the second half of the century. Flexible manufacturing allowed quick changes and modifications of old products as well as smooth introduction of new products. This radically new approach in manufacturing rose from a market that evolved so competitive that *continuous product improvement* and *research and development* became the key to survival. The early leaders in the design for manufacturing enjoyed a competitive advantage that translated into pricing. The price wars compelled all competitors to promptly adapt to the new production and marketing conditions.

Till this date, there is a sentiment that flexible manufacturing is an empirical science. A search in the literature on this area reveals that in flexible manufacturing textbooks the major concern is to expose the various machine characteristics rather than analyzing the processes themselves. As a result, production managers do not rely upon any guidelines when they design their production process. Thus, either they invest in capacity that they never utilize, or they resort to processes that are difficult to manage. In this dissertation, we address the problem of identifying the most appropriate process design for a particular environment of workorders. This

is done by comparing a variety of *flexible manufacturing systems (FMS)* that are currently employed in industry. Our study is twofold; first we examine new and existing algorithms for the management of the various FMS's currently underway, and second we use these algorithms to identify the dominant FMS for some particular scenario of workload characteristics. Thus, we produce useful guidelines for the design of manufacturing processes.

Our study sheds light to the intriguing interactions between the various factors that affect the performance of a manufacturing system. Such factors include machine characteristics, workorder specifications, machine layout, routing control, and complexity of production management. The problems treated in this dissertation are motivated from a variety of real world problems arising from PCB assembly, the glass container industry, cable manufacturing, furniture industry, fire truck and automobile assembly, etc. The manufacturing environments considered have been taken out of the operations of plants located in Florida, including the IBM plant in Boca Raton and the Emergency 1 plant in Ocala.

For our exposition we develop a framework for the *evaluation of flexibility*. A brief outline of this framework is given next.

- Identification step...
  - Identify various Modes of Flexibility
  - Identify alternative process choices
- Quantify the Value of Flexibility
  - In monetary terms (capital investment; cost analysis)
  - Performance of various process designs
    - Analytical comparison of designs
    - Average performance of designs

- Managerial Guidelines
  - Schedulability of various process designs
  - Production environment characteristics that favor one design over the other
  - Tradeoff between routing control structure and performance

This framework consists of two major steps; namely the identification and quantification of flexibility. In the former we identify the various *modes of flexibility.* We treat three such modes; namely, *dedicated machines, processing flexibility* and *routing flexibility.* In systems with processing flexibility, the machines are equipped with sophisticated tool magazines that can perform one ore more operations of the final product. Systems with routing flexibility have several machines in each production stage which prompts the intelligent balancing of the workload between the machines of each stage.

In the quantification step we quantify the *value of flexibility.* This can be done either in monetary terms or in terms of performance. Cost analysis techniques of engineering economy account for the monetary aspects of FMS's. In this dissertation we consider *machine utilization* and *throughput performance* as the major criteria for the comparison of process designs.

Multistage systems that enjoy processing flexibility can be found in PCB assembly, furniture industry, etc. For such systems we develop a fast *dynamic programming algorithm* as well as a tight *lower bound* on the optimal makespan performance. Also, we develop heuristic solutions and prove that their *worst case performance* is very good. An extensive computational experiment accompanies our algorithms. From this experiment we find out that the extra investment required for acquiring processing flexibility yields substantial performance benefits when the relative ratio of the workload of adjacent stages is either less than 0.8 or greater than 1.2. For balanced

workloads between stages processing flexibility is economically unjustified. Effectively, this becomes our first managerial guideline for process choice decisions. Even more surprising is our finding that for two-stage production, treating the flexible machines in a parallel fashion yields makespan performance which is very close to that where the flexible machines are assigned to different stages. The significance of this finding is that in a parallel layout we only need to allocate jobs to machines while in a staged layout we need both allocation of jobs to machines and scheduling of machines.

A popular class of multistage systems that enjoy routing flexibility is the class of *hybrid flowshops*. We found applications of such systems in the IBM plant in Boca Raton, FL, and in glass container industry and cable manufacturing. For two-stage production, several algorithms have been developed for hybrid flowshops over the last 20 years. We develop new heuristics that are shown to be superior to all existing algorithms in worst case as well as in average performance. In addition we develop the first algorithm for the general problem with more than 2 stages and evaluate its worst case performance. Using these algorithms we come up with several guidelines that are far from intuitive. For example, for balanced workloads investing on sophisticated tool magazines is far more profitable than investing on an additional machine. We also find the threshold workload ratios that make profitable the investment on new equipment. We summarize our results into a new concept called *design for schedulability*. This states that:

managers should balance the benefits of flexibility with ease of schedulability.

We draw an analogue of the flexibility and schedulability aspects of a production system with the *hardware and software* aspects of a computer system.

The above study applies when the resources of the manufacturing environment are machines. In labor intensive assembly lines where machines are replaced by humans, operations are subject to completely different assumptions for the coordination of

production and worker movement. The *mixed model transfer line* is such a labor intensive assembly environment which is motivated by the fire truck manufacturer in Ocala, FL, and the automobile assembly. In transfer lines we assume a *common production cycle* for all stations and *synchronous worker movement.* We compare our transfer line with the traditional mixed model assembly line that has been studied extensively in the literature. We find that the mixed model transfer line has great advantages regarding *work in process inventories, decentralized control* of manpower and process design. Based on *integer programming* and *graph theory* techniques, we develop several algorithms for minimizing the number of workers needed to operate transfer lines.

1.2 Operating Environments

In this thesis we consider manufacturing environments that are described by single machines, multistage flowshops and assembly line environments. Each of these environments is described in detail.

1.2.1 The Single Machine Model

The single machine model is the simplest and the best understood model. A set of n jobs are to be processed on a single machine or manufacturing facility. The investigation of various solution procedures and the consequences of selecting different measures of performance is easily demonstrated through this model. It can be treated as a building block for a complex model and through it the concepts of scheduling theory are better understood. From the practical point of view, there are actually some situations adequately represented by this model. For example, in the chemical industry, an entire plant can be treated as an integrated procedure that processes one product at a time, but many different products in sequence. Also, there are cases where there is a bottleneck machine in a large and complex shop where each job could consist of many different tasks and each task must be processed by a particular

machine. According to the shifting bottleneck concept, the whole shop is scheduled as if the other machines do not exist. The single machine model treated in this proposal features the following assumptions:

- A set of n independent jobs are available at time 0. Each job requires a single operation.
- The machine is continuously functioning until all jobs are completed. The machine is never kept idle.
- Either no setup time is required for each job or the setup times are sequence independent and thus can be included in each job's processing time.
- The machine cannot process more than one job at a time. Once a job starts on the machine, it cannot be interrupted until its completion.

Each job j is specified by two attributes: The processing time p_j and the due date d_j. The processing time of a job is the duration of time that job j will occupy the machine. The due date is the time at which job j is supposed to be completed. Both of these characteristics are considered to be known deterministically (usually specified by an external agency).

1.2.2 The Two-Stage Hybrid Flowshop Model

A set J of n jobs is given and every job J_i consists of two tasks, namely $(i,1)$ and $(i,2)$ with processing time requirements $p_{i,1}$ and $p_{i,2}$ respectively. All jobs are assumed to be available at time zero and no preemption is allowed for the tasks. The shop consists of two machines. The tasks $(i,1)$ must be processed at stage 1 and the tasks $(i,2)$ must be processed at stage 2. The following assumption are made:

- No task is processed by more than one machine at any time.
- No machine processes more than one job at any time.

1.2.3 The Multi-Stage Hybrid Flowshop Model

This operating environment consists of several stages. Each job has processing time requirements that are given as vectors where the k-th index corresponds to the requirements of the job on the k-th stage. The operating assumptions made for this model are identical to the assumptions of the two-stage flowshop model.

Note that this production environment includes as a special case many other environments that have been studied extensively. The hybrid flowshop with a single stage is the well known parallel identical machine environment. The k-stage hybrid flowshop environment with one machine per stage is the well known k-stage traditional flowshop. Therefore, a study of the general hybrid flowshop environment can be seen as an extension over the studies that have taken place over the last 50 years for environments such as the parallel identical machines and the traditional flowshops.

1.2.4 The Two-Stage Flexible Flowshop Model

The two stage flexible flowshop (2FFS) environment has significant processing flexibility across stages. A job can be either completely processed at one of the stages or its processing is split in a prespecified fashion between stages. For the latter case, the routing is in only one direction, from the 'upstream' stage to the 'downstream' stage. We present a pseudopolynomial dynamic programming algorithm for the solution of the problem. The 2FFS can be thought as a generalization of the two machine flowshop (2FS) and the two parallel identical machine (2P) environments.

1.2.5 The Two-Stage Flexible Jobshop Model

The two stage flexible jobshop (2FJS) environment has significant processing flexibility across stages as is the case with 2FFS. The only difference in the two environments is that 2FJS operates as a jobshop, i.e. it offers the option to schedule the first task on the 'downstream' machine with subsequent processing of the second task on the 'upstream' machine. Our objective in this model is to minimize makespan.

1.2.6 The Assembly Line Model

We consider an assembly line consisting of m stations. A set of jobs all available at time zero is given. Each job i consists of m tasks with processing time requirements $p_{i1}, p_{i2}, \ldots, p_{im}$ on stations $1, 2, \ldots, m$ respectively. The first task needs to be processed on station 1 for p_{i1} periods immediately followed by the second task which needs p_{i2} periods for processing on station 2 and so on until the m-th task is processed on the m-th station following the completion of the $(m-1)$-th task. We assume that no job is processed by more than one station and no station processes more than one task at any time. Also we require that the sequence of jobs is identical for all the stations. Although the problem is casted as an assembly line, it can represent large plants consisting of many facilities arranged in a line. Each facility can be considered as an integrated station.

1.3 Outline

In Chapter 1 we define the various production environments and outline the rest of the thesis.

In Chapter 2 we study process choice decision making problems. We define two two-stage flowshop designs which possess routing flexibility. Namely, we study the look-ahead and look-behind flowshops with minimum makespan objective. For the problem to maximize throughput for these designs, we develop fast heuristics with worst case error bound of $2 - \frac{1}{m}$. Lower bounds for the optimal makespan value are used to show that the average relative optimality gap of the heuristics is negligible. We compare the performance of our two designs with other known two stage flowshop designs with one or two machines in each stage. It is shown that the mode of flexibility possessed by a design has implications on the schedulability of the production system. This gives rise to the concept of *design for schedulability* according to which a production process should be a perfect balance between flexibility and ease

of schedulability.

In Chapter 3 we study the two-stage hybrid flowshop with minimum makespan objective. This flowshop environment that consists of multiple stages and multiple machines in each stage. These flowshops are flexible in the sense that a task within a stage can be processed by any of the machines in the stage. We refer to this design as the *hybrid flowshop*. Our objective is to schedule a set of n jobs so as to minimize makespan. The problem is $\mathcal{NP}$-complete even in the case of a single stage.

We develop a heuristic H for the two-stage hybrid flowshop that has complexity $\mathcal{O}(n \log n)$ (where n is the number of jobs) and error bound of $2 - \frac{1}{\max\{m_1, m_2\}}$ (where m_i is the number of machines at stage i, $i = 1, 2$). This bound extends a the bound developed in Chapter 2 for the case $m_1 = 1$, $m_2 = m$ and significantly improves all other results that have been developed for some special cases of the two-stage hybrid flowshop. We develop a new lower bound which is used in a computational experiment to show that the relative gap of H from optimality is small. We extend H to the case of more than two stages. We perform error bound analysis on the resulting heuristic H' whose complexity is $\mathcal{O}(kn \log n)$ (where k is the number of stages). This is the first error bound analysis for the general hybrid flowshop problem and extends the current best error bound for the traditional k machine flowshop problem.

In Chapter 4 we address a scheduling problem for a two-stage flexible flowshop (2FFS) environment with significant processing flexibility across stages and one machine per stage. A job can be either completely processed at one of the stages or its processing is split in a prespecified fashion between stages. For the latter case, the routing is in only one direction, from the 'upstream' stage to the 'downstream' stage. We present a pseudopolynomial dynamic programming algorithm for the solution of the problem. Since the 2FFS can be thought of as a generalization of the two machine flowshop (2FS) and the two parallel identical machine (2P) environments, we adapt algorithms for these problems as heuristics for the 2FFS and prove their worst case

error bound. A computational study is performed that compares the throughput performance of the 2FFS, 2FS and 2P environments, thus providing useful managerial guidelines for process choice decisions among the previously mentioned production processes.

In Chapter 5 we address a flexible jobshop scheduling problem, in order to exploit the benefits of the more complicated routing control structure of the flexible jobshop over the flexible flowshop considered in Chapter 4. We study a two-stage flexible jobshop design and compare its worst case and average performance against other two-stage flexible and non-flexible flow and jobshop designs. Depending on the workload scenario, some designs are more appropriate than others. The complexity of an optimal solution is design specific. This reveals another justification for the concept of 'Design For Schedulability'.

In Chapter 6 we study process choice decision on assembly line environments. Cellular layouts are formed by arranging several production cells in series. Group technology and cellular layouts can be combined to produce families of parts more economically than traditional process and product layouts. The advantages of cellular layouts are numerous including simplified manpower planning. We address this problem for cells having a common production cycle. Each cell is referred to as station. We refer to the design that we consider as *mixed model transfer line* and we show how this design differs from the traditional *mixed model assembly line.* After defining the problem, we present a polynomial optimal algorithm for the 2-station line. The case of 3 stations is proved to be strongly $\mathcal{NP}$-complete. Several algorithms that produce upper and lower bounds are developed for the general problem. The worst case error bound as well as average performance of these algorithms is reported. Computational results show that some of the heuristics generate near optimal solutions. Two extensions of the basic model are considered. The first, exploits the trade-off between cycle time and manpower level. The second, solves the problem to minimize the number

of utility workers, i.e. workers that are trained to work in all stations.

In Chapter 7 we conclude this thesis with suggestions on future research directions.

CHAPTER 2

DESIGN FOR SCHEDULABILITY: FLEXIBLE LOOK-BEHIND AND LOOK-AHEAD FLOWSHOPS

2.1 Introduction

In this chapter we introduce two 2-stage flowshop environments with significant flexibility within a stage. In the first design, which we refer to as *look-behind* flowshop (*LBFS*), the first stage consists of m parallel identical machines while the second stage consists of one machine only (see Figure 2.1a). The second design, which we refer to as *look-ahead* flowshop (*LAFS*), is symmetric to the LBFS since the first stage consists of a single machine while the second stage consists of m parallel identical machines (see Figure 2.1b). As we will explain later, the LBFS possesses look-behind scheduling characteristics while the LAFS possesses look-ahead scheduling characteristics.

The job set is assumed to consist of two tasks; one of which must be processed at stage 1 before the second task is processed at stage 2. Jobs entering the first stage of LBFS, are free to be allocated on any of the first stage machines and upon completion they have to be processed by the only machine of stage 2. Similarly, jobs must be processed at stage 1 of LAFS before they advance to any of the stage 2 machines. Evidently, LBFS possesses significant routing flexibility in stage 1, while LAFS possesses routing flexibility at stage 2.

Several modes of flexibility have appeared in the today's competitive manufacturing environment. To compare the advantages and disadvantages of different modes of flexibility, we consider three designs (in addition to LBFS, LAFS) representing these modes. The first design is the traditional flowshop (see Baker, 1974) depicted

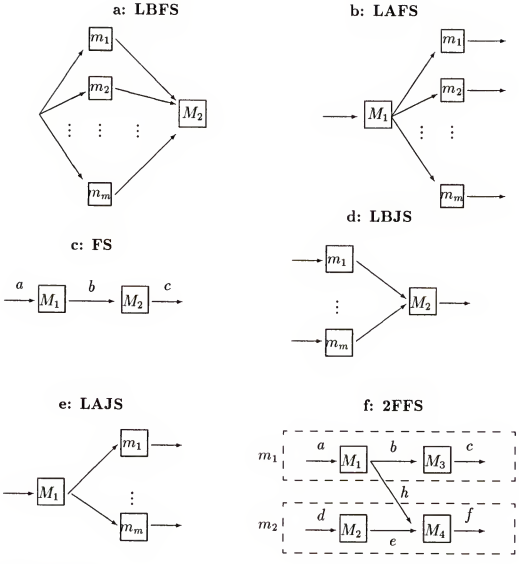


Figure 2.1. Alternative Two-Stage Flowshop Designs.

in Figure 2.1c, which we simply refer to as *FS*. Unlike LBFS, LAFS, the FS environment possesses no routing flexibility but serves as a benchmark for measuring the performance benefits incurred by adding extra machines at the multistation stage of LBFS or LAFS. Production environments where the FS design is implemented, often realize unbalanced workloads in the two stages. In the event that the management wants to speed-up production, the purchase of an extra machine becomes an appealing option, in order to alleviate the overloaded stage. Then, the question arises to find the threshold workload ratio for which the cost of the extra machine justifies the performance benefits over the existing FS design. This motivates the performance comparison of FS versus LBFS and LAFS.

The second design considered is a job shop with m machines in the first stage and one machine at the second, as in Figure 2.1d. In this design, which we refer to as *look-behind job shop* (*LBJS*), the route of each job is predetermined. Later, we justify that indeed LBJS possesses look-behind characteristics. This design treats the first stage machines in a dedicated fashion and thus does not allow for routing flexibility. Symmetrically defined is the *look-ahead job shop* (*LAJS*) (see Figure 2.1e). The LBJS and LAJS designs are the means to investigate the benefits realized by the routing flexibility of LBFS and LAFS respectively.

The third design considered in this chapter, is known as the *2-stage flexible flowshop* abbreviated by *2FFS* (see Figure 2.1f). The 2FFS environment consists of two flexible machines only, one in each of the two stages. A job can be either completely processed in one of the stages or the first task is processed by the 'upstream' stage and the second task is processed by the 'downstream' stage (see Figure 2.1f), thus preserving the flowshop structure. Evidently, 2FFS possesses both routing and processing flexibility. This design allows us to investigate the additional benefits provided by the processing flexibility mode. For all the designs as well as for LBFS and LAFS, we assume the presence of adequate storage space between the two stages.

The designs just described are frequently encountered in flexible manufacturing systems (FMS) (see Maleki, 1991 for detailed description of FMS's), where each production stage might be either a flexible machine or a flexible manufacturing cell. Flexible machines are equipped with tool magazines that can accommodate up to 160 different tools. Depending on the number and variety of tools on its magazine, a machine is able to perform a number of different operations required by a particular job. The case of relatively inflexible machines is modeled in a job shop environment where every job has to follow a particular routing of machines due to the incapability of other machines to perform operations on the job. We capture this case in the LBJS and LAJS designs.

In case that the tool magazine of the first stage machines can carry a great number of tools, the first operation of a job can be processed by either of the first stage machines. Then, the production system enjoys flexibility within the first stage. We capture this situation in the LBFS design. The symmetric case where stage 2 machines are equipped with sophisticated tool magazines is captured in the LAFS design.

In addition to stage flexibility, a system can possess processing time flexibility as in the case of 2FFS. Machines allowing for processing flexibility are accurately depicted in the flexible flowshop description as the operations of a job could be performed by either of the two machines or the operations of the job could be split between the two machines.

Flexible manufacturing cells (see Luggen, 1991) consist of a set of flexible machines, able to perform a variety of operations. Each of the manufacturing cells is a multistation line. For an environment with two flexible manufacturing cells, the LBFS problem amounts to allocating the set of jobs to the various lines of the upstream cell, in such a way that the total idle time incurred by the downstream cell is minimized. Such a requirement optimizes the throughput performance as well as the machine utilization of the production system. Similarly, the LAFS problem amounts

to allocating stage 2 tasks so that the idle time plus the workload of any of the stage 2 machines, is minimized.

The LBFS production environment considered in this chapter is a classical case of *look-behind scheduling*, since the optimal performance of the system depends upon eliminating idle time from the downstream stage, or equivalently scheduling the first stage operations so that the downstream stage does not starve. By symmetry, the LAFS is an environment with *look-ahead scheduling* characteristics. For more look-behind and look-ahead scheduling problems see Morton, 1992.

The study of scheduling problems in flexible manufacturing systems has attracted significant attention in recent years including Afentakis (1986), Erschler *et al.* (1985), Ghosh and Gaimon (1992), Herrmann and Lee (1992), Langston (1987), Lee and Herrmann (1993), Shanker and Tzen (1985), Shriskandarajah and Ladet (1986), Shriskandarajah and Sethi (1989), Stecke (1985, 1992) and Wittrock (1988), due to the importance of such systems for small-to-medium batch manufacturing.

The LBJS and LAJS designs have been studied by Lee and Herrmann (1993) and Herrmann and Lee (1992) who were motivated by problems in semiconductor testing operations. The 2FFS design has been studied by Kouvelis and Vairaktarakis (1993) who analyzed the impact of processing flexibility in flowshop production environments. The LAFS and LBFS problems have been considered before. Arthanary and Ramaswamy (1971) developed a branch and bound algorithm for the LBFS problem which can handle efficiently problem sizes of 10 or less jobs. Several heuristic algorithms without worst case error bound analysis have been developed by Gupta (1988), Gupta and Tunc (1991), Rao (1970) and others. The evaluation of these heuristics is made by establishing a lower bound and then computing the average relative gap of the heuristic solution from the lower bound. However, the lower bounds that have been developed for LBFS and LAFS are trivial and very often loose. To the best of our knowledge, the only effort for worst case error bound analysis is made

by Shriskandarajah and Sethi (1989) for the LAFS problem. However, the bounds obtained are big (in the neighborhood of 300%) thus indicating that the LBFS and LAFS problems are very difficult.

We conclude that the LBFS and LAFS systems have long but relatively unsuccessful precedence in the literature. In this chapter we develop new lower bounds and heuristics for LBFS and LAFS. We show that our heuristics have near optimal average performance and that their worst case error bound is small. We use these heuristics for performance comparisons of the various designs treated in this chapter. These comparisons are motivated by eminent managerial dilemmas regarding the value of flexibility, as described earlier. Our motivation for this chapter is primarily the comparison of the various flexible designs of Figure 2.1 and secondarily the study of the LBFS and LAFS problems.

This chapter consists of two parts. The first part which consists of Sections 2.2, 2.3 focuses on the LBFS and LAFS problems. In particular, in Section 2.2 we introduce the designs LBFS and LAFS and in Section 2.3 we present fast heuristic algorithms with very good worst case performance. In addition, we establish lower bounds which we use in a computational experiment to report the average relative gap of the heuristic solutions from the lower bound.

The second part of the chapter exploits the benefits incurred by the various modes of flexibility. In particular, in 2.4.1 we find the range of the performance ratios of the design LBFS (LAFS) versus FS, LBJS (LAJS) and 2FFS. These theoretical results suggest that the various designs behave differently under a variety of environment characteristics. Thus, in 2.4.2 we perform an experiment that finds the average performance of each design under different job characteristics. In light of these comparisons, for balanced workloads investing in tool magazines is more beneficial than investing in additional equipment. Also, it is shown that the mode of flexibility possessed by a design has implications on the schedulability of the production system. This gives

rise to the concept of *design for schedulability* according to which a production process should be a perfect balance between flexibility and ease of schedulability. We summarize our results with a set of guidelines useful to a manager who makes process choice decisions, in Section 2.5. In section 2.6 we give an analog of the flexibility and schedulability of production systems. We conclude this chapter in Section 2.7.

2.2 The Flexible LBFS and LAFS Designs

Let J be a set of n jobs. A job $j_i \in J$ consists of two tasks; namely $(i,1)$ and $(i,2)$ with corresponding processing time requirements p_{i1} and p_{i2}. We define a 2-stage flexible flowshop environment to process the job set J as follows: The first stage consists of m parallel identical machines $m_1, m_2, \ldots, m_m$, each of which can process task $(i,1)$ only. The first stage task of j_i can be processed by either of the m first stage machines. The second station consists of a single machine that must process all the second stage tasks $(i,2)$ and therefore there is no routing choice for the $(i,2)$ tasks of J (see Figure 2.1a). Due to the flowshop constraint, the sequence of jobs on the first stage must be identical to the sequence of jobs in the second stage. We refer to this flowshop problem as LBFS because it is a flowshop with look-behind characteristics. A symmetric production environment consisting of a single machine in stage 1 and m parallel identical machines at stage 2 is called LAFS because it is a flowshop design with look-ahead characteristics (see Figure 2.1b).

The LBFS and LAFS configurations are very common subsystems of many sectors in industry. Examples of applications include semiconductor manufacturing (see Herrmann and Lee, 1992; Lee and Herrmann, 1993), glass container industry (see Paul, 1979), cable manufacturing (see Narasimhan and Panwalker, 1984) and others. The LBFS and LAFS designs are a generalization of the m parallel identical machines environment (mP) where it is assumed that $p_{i2} = 0$ for every $j_i \in J$ in LBFS and $p_{i1} = 0$ for every $j_i \in J$ in LAFS. The mP environment has been well studied over the

last 25 years by several researchers. A review including most results in this area is given by Cheng and Sin (1990). Garey and Johnson (1979) have shown that minimizing makespan in the mP environment is $\mathcal{NP}$-complete. Since the mP environment is a special case of LBFS and LAFS, minimizing makespan in these environments is $\mathcal{NP}$-complete as well. This observation directs our attention to heuristic algorithms. This is the focus of the next section.

2.3 Heuristic Algorithms for LBFS and LAFS

In this section, we focus our discussion to the LBFS design. All theory developed, however, is easily extendable to the LAFS design. Therefore, we only provide formal proofs for the LBFS case and summarize the corresponding results for the LAFS.

Since the *LBFS* design is a hybrid between the mP environment and the traditional two machine flowshop FS, we are motivated to develop heuristic algorithms that are in a certain sense hybrids of existing good algorithms for these two environments. Minimizing the makespan in FS is solved in $\mathcal{O}(n \log n)$ time by Johnson's algorithm which we refer to as *JA* (see Johnson, 1954). The JA algorithm follows immediately from the rule: *Job* j_i *precedes job* j_j *if* $\min\{p_{i1}, p_{j2}\} \leq \min\{p_{j1}, p_{i2}\}$.

The generic form of heuristic algorithms for the mP environment (also known as list scheduling heuristics) that are proved to have very good performance (see Graham, 1966) is given next.

List Scheduling Heuristic for m*P:*

1. Let S be an order of jobs in J.
2. Pick the job at the top of the list S and load it to the first available machine.
3. If S is empty stop, else goto Step 2.

According to the first available machine rule (*FAM*), the job to be scheduled next in the mP environment, is assigned to the first machine that becomes available, i.e.,

the machine that finishes first the job (if any) previously assigned to it. Depending on the order S, the above heuristic produces different solutions. Thus, a first question in the quest for heuristic solutions for LBFS and LAFS is to evaluate the performance of list scheduling heuristics. Towards this end, Shriskandarajah and Sethi (1989) showed that if S is an arbitrary list of jobs, then a list scheduling heuristic based on the FAM rule has worst case error bound of $3 - \frac{1}{m}$ for the LBFS problem. It is easy to extend this heuristic for the LAFS problem and still obtain the bound of $3 - \frac{1}{m}$. In light of these results, we are interested in finding heuristics that have better worst case performance. The following heuristic uses the Johnson's order in conjunction with the FAM rule.

Heuristic H_1:

1. Apply JA with respect to the processing times $\{(\frac{1}{m}p_{i1}, p_{i2}) : i = 1, 2, \ldots, n\}$. Let S be the resulting sequence.

2. Pick the first job in S and load it to the first available machine at station 1. Repeat Step 2 until all jobs in S are scheduled.

3. Schedule jobs in M_2 according to sequence S.

At Step 1 of H_1 a sequence S of jobs is produced, at Step 2 an assignment of $(i, 1)$ tasks on the first stage machines is made and at Step 3 idle time is inserted appropriately so that the jobs in M_2 are scheduled according to sequence S as soon as possible without violating the flowshop constraints. Since JA requires $\mathcal{O}(n \log n)$ time, this is also the computational effort required by H_1.

The next lemma will be used to find the error bound of H_1. For this lemma we introduce the following *auxiliary* problem. Replace the first stage machines $m_1, m_2, \ldots, m_m$ by a dummy machine M_1 and replace the processing time requirement of p_{i1} units on one of $m_1, m_2, \ldots, m_m$ by the requirement of $\frac{1}{m}p_{i1}$ units on M_1. No change is made

in the processing time requirements of $(i,2)$ tasks. This way we define an auxiliary 2-machine flowshop problem on M_1, M_2. We will refer to this problem as *AFS*.

Lemma 1 *Let C_{AFS} be the completion time found by JA for the auxiliary problem. Then, $C_{AFS} \leq C_{LBFS}$.*

Proof. Let S^* be an optimal sequence for LBFS (of course the processing times for this problem are $(p_{i,1}, p_{i,2})$). Let $S = \{j_1, \ldots, j_n\}$ be the set of jobs ordered according to the nondecreasing order of completion times of the tasks $(i,1)$ on the first stage machines. Then, $t_{j_i} = t_i$ and $t_1 \leq t_2 \leq \ldots \leq t_n$.

Consider the auxiliary 2-machine flowshop problem defined earlier. Schedule the jobs in AFS according to sequence S. Then, for every $1 \leq i \leq n$, $t_i \geq \frac{1}{m}\sum_{k=1}^{i} p_{k1}$ because all of the tasks $(1,1), \ldots, (i,1)$ are completed by t_i in the schedule S^* for LBFS. Thus, every $(i,1)$ task of AFS finishes no later than it does on LBFS. Equivalently, due to the flowshop constraints, the idle time preceding task $(i,2)$ in LBFS is at least as much as the idle time preceding $(i,2)$ in AFS.

Hence, if C_S is the makespan of sequence S for the AFS problem, we have $C_S \leq C_{LBFS}$. However, the sequence S is not necessarily optimal for the AFS problem and hence $C_{AFS} \leq C_S$. The last two inequalities show that $C_{AFS} \leq C_{LBFS}$. This completes the proof of the lemma. □

Let C_{H_1} be the makespan of the schedule obtained by H_1. Then,

Theorem 1 $\frac{C_{H_1}}{C_{LBFS}} \leq 2 - \frac{1}{m}$ *and the bound is tight.*

Proof. Let S be the Johnson's order for the AFS problem and S_H the sequence obtained by H_1 for the LBFS problem. In S_H, task $(i,1)$ finishes no later than $\frac{1}{m}\sum_{k=1}^{i} p_{k1} + \frac{m-1}{m} p_{i1}$ because the FAM rule is used for allocating $(i,1)$ tasks to the first stage machines. Therefore, the idle time forced by the task $(i,1)$ on M_2 of LBFS is no more than $\frac{m-1}{m} p_{i1}$ periods more than the idle time forced by $(i,1)$ on the M_2

machine of AFS.

Since the idle time inserted for task $(i,1)$ on M_2 of LBFS shifts to the right all subsequent tasks $(i+1,2),\ldots,(n,2)$, M_2 of LBFS can have at most $\frac{m-1}{m}p_{\max}$ periods of idle time more than AFS does, where $p_{\max} = \max_i p_{i1}$. Equivalently, $C_H \leq C_{AFS} + \frac{m-1}{m}p_{\max}$. By Lemma 1, $C_{AFS} \leq C_{LBFS}$. Also, $p_{\max} \leq C_{LBFS}$ and hence $C_H \leq (2-\frac{1}{m})C_{LBFS}$.

To see that the bound is tight, consider the case where $p_{i2} = 0$. Then, the Johnson's algorithm, when applied to AFS, can output any sequence depending on how ties are broken. Then, H_1 reduces to the RDM heuristic (a heuristic where the list S is a random permutation of the set of jobs) for mP which is known to have a tight error bound of $2-\frac{1}{m}$ (see Graham, 1966). This completes the proof of the theorem. □

The corresponding results for the LAFS design, are as follows:

Heuristic H_2:

1. Apply JA with respect to processing times $\{(p_{i1}, \frac{1}{m}p_{i2}) : i = 1,2,\ldots,n\}$. Let S be the resulting sequence.

2. Schedule jobs in M_1 according to sequence S.

3. Pick the job j_i at the top of S and load it to the first available machine at station 2, but not earlier than the completion time of the $(i,1)$ task. Repeat Step 2 until all jobs in S are scheduled.

Theorem 2 $\frac{C_{H_2}}{C_{LAFS}} \leq 2-\frac{1}{m}$ *and the bound is tight.*

The above theorem shows that the heuristic H_2 which is based on the Johnson's sequence with respect to $\{(p_{i1}, \frac{1}{m}p_{i2}) : i = 1,2,\ldots,n\}$ has a worst case error bound of $2-\frac{1}{m}$. This improves the current best known results of Shriskandarajah and Sethi (1989) for the the LAFS problem. They show that a heuristic algorithm based on an

arbitrary ordering of the jobs has worst case error bound of $3-\frac{1}{m}$. Also, they show that a heuristic based on the Johnson's sequence with respect to $\{(p_{i1}, p_{i2}) : i = 1, 2, \ldots, n\}$ has error bound of $3 - \frac{3}{m} + \frac{1}{m^2}$. To the best of our knowledge, no error bound results are known in the literature for the LBFS problem.

We would like to compute the average deviation of H_1 from the optimal solution of LBFS and the average deviation of H_2 from the optimal solution of LAFS. To do this, however, we need to develop relatively fast branch and bound algorithms that solve LBFS, LAFS optimally. In our opinion, this is rather unlikely. The reason is that there are m^n different ways of allocating jobs to the multistation stage. Given such an allocation, the scheduling problem reduces to a look-behind job shop (LBJS) and look-ahead job shop (LAJS) respectively. In these environments, each of the machines of the multistation stage are dedicated to a subset tasks. Both of the LBJS and LAJS problems have been shown to be $\mathcal{NP}$-complete in the strong sense even in the case where the multistation stage has 2 machines only, see Herrmann and Lee (1992), Lee and Herrmann (1993). Since the LBFS and LAFS problems do not seem to enjoy any interesting properties that eliminate most of the m^n possible allocations for the multistation stage, any exhaustive algorithm for these problems is doomed. For these reasons, we perform a computational experiment where we compute the average relative gap of H_1, H_2, from the lower bound C_{AFS} of Lemmata 1,2. This can be done very efficiently since all of the values C_{AFS}, C_{H_1}, C_{H_2} can be computed in polynomial time.

2.3.1 Average Performance of Heuristics

In Table 2.1 we report the relative gap $\frac{C_H - C_{AFS}}{C_{AFS}} 100\%$ where H is any of the heuristics H_1 and H_2. In this experiment we report the average and worst case relative gap for different problem sizes and processing time ratios. We report results for problem sizes with $m = 2$, 3, 4 stage 1 machines and 20, 30 and 40 jobs. We

experimented with the processing time ratios 1 : 1, 2 : 1, 3 : 1, 4 : 1 and their reciprocals. For example, a ratio 2 : 1 means that on the average the processing time of a task $(i,1)$ is double the processing time of a task $(i,2)$. Also, the processing time of $(i,1)$ is chosen randomly from a uniform distribution on $[1,10]$ and the processing time of $(i,2)$ is chosen randomly from a uniform distribution on $[1,20]$. For each combination, we find the average relative gap over 50 randomly selected problems. A similar experiment is made for the LAFS problem.

When the second stage is the overloaded one, the average relative optimality gap of H_1 is very rarely different than 0. This is because in case that stage 2 is overloaded, the stage 2 machine of LBFS is the bottleneck and therefore the advantage of having many machines in stage 1 of LBFS does not alleviate the bottleneck operations; just like what happens in the auxiliary problem AFS. As a result, the AFS and LBFS problems end up having the same makespan. In the symmetric case where stage 1 is overloaded, the performance of H_2 for LAFS has similar performance. However, having many machines at stage 1 when stage 2 is overloaded, or having many machines at stage 2 when stage 1 is overloaded, do not justify the use of LBFS and LAFS respectively. In both cases, the average relative gap of H_1 for LBFS and H_2 for LAFS is 0.0%. For this reason, we do not report these cases in Table 2.1.

In the important case where stage 1 is overloaded in LBFS, the heuristic H_1 has an average relative gap of 0.72%. This gap reduces as the number of jobs increases and as the number of stage 1 machines increases. Also, the average relative gap increases as the workload ratio $\frac{\sum_i p_{i1}}{\sum_i p_{i2}}$ increases. The size of the relative gap along with the fact that the corresponding deviation from optimality is at most as big, establishes H_1 as an exceptional tool for LBFS. In the other important case where stage 2 is overloaded in LAFS, the heuristic H_2 has an average relative gap of 0.31%. Again, we feel confident in recommending H_2 as an adequate tool for scheduling operations in the LAFS environment.

Table 2.1. Relative Gap of H_1, H_2 from the Lower Bound C_{AFS}.

$\frac{C_{H_1}-C_{AFS}}{C_{AFS}}100\%$												
m	2	3	4	2	3	4	2	3	4	2	3	4
size/ratio	2 : 1			2 : 1			3 : 1			4 : 1		
20	0.0	0.0	0.0	1.2	0.3	0.5	2.7	2.9	2.1	3.1	6.8	5.2
30	0.0	0.0	0.0	0.8	0.2	0.1	2.0	1.8	0.3	2.3	4.8	3.5
40	0.0	0.0	0.0	0.4	0.0	0.1	1.6	1.2	0.1	1.6	4.0	2.0
Average	0.0	0.0	0.0	0.8	0.4	0.2	2.1	0.6	0.5	2.1	1.2	0.8
$\frac{C_{H_2}-C_{AFS}}{C_{AFS}}100\%$												
	1 : 1			1 : 2			1 : 3			1 : 4		
20	0.0	0.1	0.2	0.2	0.5	0.4	0.7	0.6	0.8	0.8	1.8	2.1
30	0.0	0.0	0.0	0.0	0.2	0.1	0.1	0.4	0.4	0.3	0.8	0.7
40	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.3	0.1	0.1	0.5	0.4
Average	0.0	0.0	0.0	0.0	0.2	0.2	0.2	0.4	0.4	0.4	1.0	1.0

This section concludes the study of solution algorithms for the LBFS and LAFS designs. Starting from the next section, we will compare LBFS and LAFS with various designs that possess different modes of flexibility.

2.4 Design for Schedulability

In the remaining of this chapter we investigate the performance benefits of each design as a function of the mode of flexibility possessed by the design. The term *design* refers to the *machine layout design* and not to the *product design* as is usually implied in literature. From our study in this section, it will be apparent that the schedulability (i.e. the ease of finding an optimal or near optimal schedule) of the various designs differs significantly. Therefore, a manager who has to make process choice decisions should find the perfect balance between performance benefits and schedulability of the design. As of today, the latter component has been underestimated and very often forgotten.

Next we compare the advantages and disadvantages of various 2-stage flowshops each possessing a different mode of flexibility than LBFS and LAFS. We do two

sorts of comparisons. Namely analytical comparison where we find the range of the performance ratio of one design over the other, and average performance comparisons where we experiment on the effect of important characteristics of the production environment on the performance of the designs. The former set of comparisons takes place in subsection 2.4.1 and the latter in 2.4.2.

2.4.1 Analytical Comparisons

We would like to know the range of the performance ratio $\frac{C_{LBFS}}{C_O}$ where C_O is the makespan of FS, or LBJS, or 2FFS. Similar ratios for LAFS are also desirable. Our formal discussion in this subsection, will refer to the LBFS design; however, similar arguments can be made for LAFS. The analytical comparisons to be performed are

- look-behind flowshop versus traditional flowshop,
- look-behind flowshop versus look-behind job shop,
- look-behind flowshop versus flexible flowshop,
- look-ahead flowshop versus traditional flowshop,
- look-ahead flowshop versus look-ahead job shop,
- look-ahead flowshop versus flexible flowshop.

The FS design: unidirectional routing

The FS design described in Section 2.1 serves as a benchmark for comparing the additional performance benefits of increasing the number of machines in the first stage of the FS design. In FS, routing is only in the direction of M_1 to M_2 and scheduling is very simple as given by the Johnson's algorithm. On the contrary, both routing and scheduling are considerably more difficult in LBFS; however, the makespan benefits of this design are more pronounced. The performance ratio of LBFS and LAFS versus

FS are given in the next theorem.

Theorem 3 $1 \leq \frac{C_{FS}}{C_{LBFS}} \leq m$ *and* $1 \leq \frac{C_{FS}}{C_{LAFS}} \leq m$ *and these performance bounds are tight.*

Proof. Consider the first double inequality. Since FS is a subsystem of LBFS, we have that $C_{LBFS} \leq C_{FS}$. Let S^* be an optimal sequence for the LBFS problem. S^* partitions the job set J into the subsets $J_1, \ldots, J_m$ where J_i is the subset of jobs processed by m_i, $i = 1, 2, \ldots, m$. Let S_i be the Johnson's sequence for J_i and C_i the corresponding makespan. Since the sequence $S_1 S_2 \ldots S_m$ is feasible for FS, we have that $C_{FS} \leq \sum_{i=1}^{m} C_i \leq m C_{LBFS}$ and hence $\frac{C_{FS}}{C_{LBFS}} \leq m$.

To show that the bounds of the first double inequality are tight, consider the problem with $p_{i1} = 0$ for $i = 1, 2, \ldots, n$ and p_{i1} arbitrary. In this case $C_{FS} = C_{LBFS} = \sum_i p_{i2}$ in which case the left hand side bound is tight. For the right hand side bound let $p_{i1} = 0$ and $p_{i1} = k$ for $i = 1, 2, \ldots, m$. Then, $C_{LBFS} = k$ while $C_{FS} = mk$ in which case $\frac{C_{FS}}{C_{LBFS}} = m$.

Similar arguments establish the validity and tightness of the second double inequality. This completes the theorem. □

The LBJS design: machines are dedicated

The LBJS environment with 2 machines at stage 1 has been studied by Lee and Herrmann (1993). In their study, a set of jobs $J = \{j_1, j_2, \ldots, j_n\}$ is given and every job j_i consists of two tasks, namely $(i, 1)$ and $(i, 2)$ with processing time requirements p_{i1} and p_{i2} respectively. All jobs are assumed to be available at time zero and no preemption is allowed for the tasks. The set J of jobs is partitioned into a set J_a of jobs that must follow the route a-b-e, and the set J_b of jobs that must follow the route c-d-e (see Figure 2.1d). This assumption treats machines M_a, M_b in a dedicated fashion. We refer to this production environment as *LBJS* because it is a job shop with look-behind characteristics.

It is evident that the only difference between LBJS and LBFS is that the latter possesses routing flexibility while former does not. Clearly, the throughput performance of LBFS is no worse than the performance of LBJS. The added flexibility within stage 1 of LBFS can be achieved either by sophisticated tooling of machines M_a, M_b so that they can process any job, or by replacing the first stage machines of LBJS with flexible machines that can process all jobs. In this subsection we find the throughput performance ratio of LBFS as compared with LBJS. Let C_{LBFS} be the optimal makespan value for LBFS and C_{LBJS} the optimal value for the LBJS environment. The next theorem gives the range of values of the performance ratios of LBJS, LAJS over LBFS, LAFS respectively.

Theorem 4 $\frac{C_{LBJS}}{C_{LBFS}} \leq m$ *and* $\frac{C_{LBJS}}{C_{LAFS}} \leq m$ *and these performance bounds are tight.*

Proof. To prove the first inequality suppose that S^* is an optimal schedule for LBFS. We will construct a schedule S for LBJS with makespan no more than mC_{LBFS}. Indeed, let J_i be the subset of J that must be processed by m_i of LBJS, $i = 1, 2, \ldots, m$. The, construct S as follows.

Schedule in S the machine m_i of LBJS to process all stage 1 tasks of J_i that are not processed by the machine m_i of LBFS, for $i = 1, 2, \ldots, m$. With this assignment, the m_i machine of LBJS can process tasks that are processed by anyone of $m - 1$ machines of LBFS. Therefore, the LBJS needs no more than $(m-1)C_{LBFS}$ time to process all the above tasks. Starting at time $(m-1)C_{LBFS}$, schedule S to process on m_i the stage 1 tasks of J_i that are processed by the machine m_i of LBFS, for $i = 1, 2, \ldots, m$. Also, schedule all the stage 2 tasks to be processed by the M_2 machine of LBJS. Then, S need no more than C_{LBFS} time to execute the latter set of tasks and therefore the makespan of S is no more than mC_{LBFS}. It is easy to verify that the S is a feasible schedule for LBFS. Since S is not necessarily optimal for LBJS, we have that $C_{LBJS} \leq mC_{LBFS}$.

It is apparent that the above bound is driven by the subsets J_i, $i = 1, 2, \ldots, m$. A choice of J_i's that results to unbalanced workloads for the various machines can have very negative impact on the performance of LBJS. Consider for instance the case where $p_{i1} = k$ and $p_{i2} = 0$ for $i = 1, 2, \ldots, m$. Then, $C_{LBFS} = k$. If $J_1 = J$ and $J_r = \emptyset$ for $r = 2, \ldots, m$, we have that $C_{LBJS} = mk$, i.e. $\frac{C_{LBJS}}{C_{LBFS}} = m$. This example shows that the performance bound of m is tight.

Similar arguments establish the validity and tightness of the second inequality. This completes the theorem. □

The 2FFS design: flexibility across the production line

We consider a configuration that has been studied by Kouvelis and Vairaktarakis (1993) known as the two stage flexible flowshop (2FFS). This configuration is defined as follows: A set of jobs $J = \{j_1, j_2, \ldots, j_n\}$ is given and every job j_i consists of two tasks, namely $(i, 1)$ and $(i, 2)$, with processing time requirements p_{i1} and p_{i2} respectively. All jobs are assumed to be available at time zero and no preemption is allowed for the tasks. The jobs in J are to be processed in a two stage flowshop which has the following type of processing flexibility. Every job j_i can be processed entirely on the upstream stage m_1 for $p_i = p_{i1} + p_{i2}$ periods or entirely in the downstream stage m_2 for p_i periods or task $(i, 1)$ is processed on m_1 and task $(i, 2)$ is processed on m_2. For maximizing the throughput, as well as the machine utilization of the system, we are interested in sequencing the jobs in J so that the resulting schedule on the two stage flexible flowshop minimizes the makespan. This design is depicted in Figure 2.1f and we refer to it as *2FFS*. In the figure, every job can follow one of the three possible routes a-b-c, a-h-f or d-e-f.

Due to its routing flexibility, 2FFS possesses flexibility across the production line. Also, 2FFS possesses processing flexibility because a job can be processed entirely on one of the machines m_1, m_2, or it can split its processing time between m_1, m_2, in a prespecified fashion. Note that the LBFS design possesses significant routing

flexibility and no processing time flexibility. Also, the 2FFS design, which possesses processing time flexibility, also possesses limited routing flexibility.

Scheduling the 2FFS environment is not an easy task. Kouvelis and Vairaktarakis (1993) presented a pseudopolynomial dynamic programming algorithm of complexity $\mathcal{O}(nMS^3)$, that solves 2FFS optimally. We will investigate the throughput performance of 2FFS as compared to that of LBFS.

In the next theorem we find the range of the throughput ratio of 2FFS versus LBFS. We assume that C_{2FFS} and C_{LBFS} are the optimal makespan values for 2FFS and LBFS, respectively.

Theorem 5 $\frac{1}{\lceil \frac{m+1}{2} \rceil} \leq \frac{C_{LBFS}}{C_{2FFS}} \leq 2$ *and* $\frac{1}{\lceil \frac{m+1}{2} \rceil} \leq \frac{C_{LAFS}}{C_{2FFS}} \leq 2$ *and these performance bounds are tight.*

Proof. We consider the first double inequality. For the left hand side of this inequality assume that S^* is an optimal schedule for LBFS. We will construct a schedule S for 2FFS with makespan no more than $\lceil \frac{m+1}{2} \rceil C_{LBFS}$. Indeed, let J_i be the subset of jobs in J processed by m_i in S^*, for $i = 1, 2, \ldots, m$. To construct S, process the $(i,1)$ tasks of $J_3 \bigcup J_5 \bigcup \ldots$ by the m_1 machine of 2FFS and the $(i,1)$ tasks of $J_2 \bigcup J_4 \bigcup \ldots$ by m_2. Since $\sum_{j \in J_i} p_{j1} \leq C_{LBFS}$ for $i = 2, 3, \ldots, m$, the execution of the $(i,1)$ tasks of $J_2 \bigcup J_3 \bigcup \ldots \bigcup J_m$ takes no more than $\lceil \frac{m-1}{2} \rceil C_{LBFS}$ time in S. Following these tasks is S, schedule the $(i,1)$ tasks of J_1 on m_1 and the $(i,2)$ tasks of J on m_2. The set of tasks is executed in no more than C_{LBFS} time and hence S has makespan no more than $(\lceil \frac{m-1}{2} \rceil + 1) C_{LBFS}$. It is easy to verify that S is a feasible sequence for 2FFS. Since S is not necessarily optimal for 2FFS, we have that $C_{2FFS} \leq \lceil \frac{m+1}{2} \rceil C_{LBFS}$. To show that this bound is tight, consider the problem where $p_{i1} = k$ and $p_{i2} = 0$ for $i = 1, 2, \ldots, m$ and $p_{m+1,1} = 0$, $p_{m+1,2} = k$. Then, $C_{LBFS} = k$ while $C_{2FFS} = \lceil \frac{m+1}{2} \rceil k$ and hence the bound is tight.

For the right hand side of the inequality assume that Q^* is an optimal schedule

for 2FFS. We will construct a schedule Q for LBFS with makespan no more than $2C_{2FFS}$. In Q, let m_1 of LBFS process all stage 1 tasks that are processed by the machine m_1 of 2FFS in Q^*. For this set of tasks, Q requires no more than C_{2FFS} time. After the completion of the above tasks, schedule on the m_1 machine of Q, the stage 1 tasks that are processed by the m_2 machine of 2FFS. Also, schedule on the m_2 machine of Q, the stage 2 tasks that are processed by the m_2 machine of 2FFS. Then, Q needs no more than C_{2FFS} time to execute the extra tasks. Since Q is not necessarily optimal for LBFS, we have that $C_{LBFS} \leq 2C_{2FFS}$.

The above argument capitalizes on the fact that the LBFS environment has one machine at stage 2, while the 2FFS can use both of its machines to process stage 2 tasks. This becomes apparent at the following problem that shows that the bound of 2 is tight. Let $p_{i1} = 0$ and $p_{i2} = 1$ for $i = 1, 2, \ldots, n$, and $p_{n+1,1} = 0$, $p_{n+1,2} = n$. Then, $C_{LBFS} = 2n$ while $C_{2FFS} = n$. This completes the theorem for the first double inequality.

Similar arguments establish the validity and tightness of the second double inequality. This completes the theorem. □

The theory developed in this section suggests that depending on the workload scenario, the LBFS and LAFS designs may provide significant or minor benefits as compared with other designs (in the case of 2FFS, the LBFS and LAFS designs might even be inferior). Then, it is essential to investigate the workload scenarios that favor each design. According to this investigation, we would like to come up with guidelines for the production manager who wants to choose the design that suits his/her objectives the most. This is the focus of the next subsection.

2.4.2 Average Comparisons

In view of the results of the last subsection, we want to identify the workload scenarios for which one design outperforms the other. For this reason we perform computational experiments in which we tabulate the performance of the various designs under a variety of environment characteristics. To make this experiment possible, we focus our attention to designs that have one or two machines in each stage. In particular, we consider the LBFS and LAFS designs with two machines at stage 1 and stage 2 respectively. In the rest of the paper when we refer to the LBFS environment we will mean the case with only two machines at stage 1. Similarly, by LAFS we will refer to the special case where stage 2 consists of two machines only.

The tables that follow report the performance of LBFS and LAFS versus FS and 2FFS. The LBJS and LAJS designs are disregarded in this experiment because, as we saw in the last section, their performance depends on the choice of the subset J_i of jobs to be processed by m_i, $i = 1, 2, \ldots, n$. However, in comparing the performance of the various designs, we face the following difficulty. The 2FFS design considered in the next section, accepts, as we mentioned earlier, an optimal solution produced by a dynamic programming algorithm, in pseudopolynomial time. When comparing the performance of LBFS, LAFS versus 2FFS, it is unfair to consider polynomial time heuristics for LBFS, LAFS and exact pseudopolynomial algorithms for 2FFS. This is more relevant for the LBFS problem since the performance of H_2 for the LAFS problem is extremely good. To achieve some fairness in our computations, we will develop a pseudopolynomial time algorithm for the LBFS problem, which although not optimal, it improves upon the performance of the heuristic H_1 in Section 2.3.

Dynamic programming based algorithm for LBFS

It is very unlikely to discover optimal dynamic programming algorithms for the LBFS and LAFS designs with minimum makespan objective. The reason is that the

optimal solution of these problems does not possess any obvious ordering of the jobs in the multistation stage. In particular, the Johnson's order is not necessarily optimal for these problems, as has been demonstrated in Herrmann and Lee (1992) and Lee and Herrmann (1993).

The rest of our discussion refers to the LBFS design. Similar arguments will be made for LAFS. Consider an optimal solution for the LBFS with the additional constraint that the jobs are ordered according to Johnson's order with respect to $(\frac{1}{m}p_{i1}, p_{i2})$ on the single machine of stage 2. As we saw in Section 3, the solution of H_1 possessed this ordering property. The difference between H_1 and the above proposed algorithm is that the former assigns the first stage tasks according to the FAM rule while the latter assigns those tasks optimally. Therefore, the error bound of $2 - \frac{1}{m}$ is also an error bound for the proposed algorithm.

As we reported in Section 2.3, the average relative gap of the solution provided by H_1, from the lower bound C_{AFS}, is very small. Since the algorithm to be developed will always dominate the performance of H_1, we expect it to perform near optimally. With this motivation, we proceed to describe a dynamic programming algorithm *(DP)* that finds an optimal solution for LBFS subject to the constraint that the jobs are ordered in stage 2 according to Johnson's order with respect to $(\frac{1}{m}p_{i1}, p_{i2})$, say $S = \{j_1, \ldots, j_n\}$.

To compute the makespan of a solution for DP, it is enough to know the total idle time of the stage 2 machine, say id. Then the makespan produced by the solution would be $id + \sum_{i=1}^{n} p_{i2}$. We can always assume that the stage 2 tasks are shifted right so that the stage 2 machine is continuously busy after the commencement of j_1. Therefore, for a particular value of id the order S induces a starting time $d_i = id + \sum_{j=1}^{i-1} p_{j2}$ for each task $(i, 2)$ at stage 2. Effectively, d_i becomes a deadline for the completion of task $(i, 1)$. Therefore, the dynamic program only needs to verify the existence or not of a feasible schedule for the mP environment of stage 1. Equivalently,

our problem reduces to a parallel identical machine problem with maximum lateness objective $L_{\max} = \max_i\{C_i - d_i\}$, where C_i is the completion time of task $(i,1)$. If $L_{\max} \leq 0$, then a feasible schedule exists. Otherwise, the idle time inserted in stage 2 is not enough and hence id should be increased.

The problem to minimize $L_{\max}$ with release dates r_i and due dates d_i, has been considered by Carlier (1987) who developed a branch and bound algorithm, and Jackson (1955) who investigated heuristic algorithms. Our problem however, is a special case where $r_i = 0$ for every j_i and as we will show enjoys an interesting DP formulation.

Let $\{j_1, j_2, \ldots, j_n\}$ be the Johnson's order with respect to $(\frac{1}{m}p_{i1}, p_{i2})$. Then, define the following:

$A_i = \sum_{j=1}^{i} p_{i1}$ for $i = 1, 2, \ldots, n$ and

$f_i(t) = \begin{cases} 1 & \text{if there exists a schedule whose total processing time on } m_1 \text{ of jobs in} \\ & \{j_1, j_2, \ldots, j_i\} \text{ is } t, \text{ and every job } j \in \{j_1, j_2, \ldots, j_i\} \text{ finishes before } d_j \\ 0 & \text{otherwise.} \end{cases}$

Boundary Conditions:

$f_0(0) = 1$ and

$f_0(t) = 0$ otherwise.

Recurrence Relation:

$$f_i(t) = \max \begin{cases} f_{i-1}(t - p_{i1}) & \text{if } t \leq d_i \\ f_{i-1}(t) & \text{if } A_i - t \leq d_i \\ 0 & \text{otherwise.} \end{cases}$$

Solution:

Let f_n^* denote the solution to the dynamic program with the above given boundary conditions. Then,

$$f_n^* = \max_t f_n(t)$$

Justification:

The above dynamic programming algorithm utilizes the binary variables $f_i(t)$. At the i-th stage of the dynamic program, job j_i has to be scheduled. The first two

branches of the recursive relationship correspond to machines m_1, m_2 respectively. If $t - p_{i1} \leq d_i$, then scheduling j_i next on m_1 does not violate the deadline of j_i. If in addition $f_{i-1}(t - p_{i1}) = 1$ then appending j_i on m_1 produces a feasible schedule for the first i jobs of S.

In case that the total workload on m_1 is t, the workload on m_2 is $A_{i-1} - t$. Therefore, scheduling j_i next on m_2 is feasible only in case that $A_i - t \leq d_i$ and $f_{i-1}(t) = 1$. Otherwise, the last branch assigns the value 0 to $f_i(t)$ which means that no feasible schedule exists with total processing time on m_1 of jobs in $\{j_1, j_2, \ldots, j_i\}$ equal t.

Note that the idle time preceding the first job of the second stage machine of LBFS does not exceed $MS = \sum_{i=1}^{n} p_{i1}$. Then by applying bisection search in the range of id, we need no more than $\log MS$ iterations of the above dynamic program to find the smallest value of id for which a feasible schedule exists. We refer to the binary search based on the above dynamic programming algorithm as *DP1*.

Complexity of DP1:

It is evident that $t \leq MS$ and i takes on values $1, 2, \ldots, n$. Hence the number of states of the dynamic program is nMS. For each state it takes $\mathcal{O}(1)$ effort to evaluate the recurrence relationship and therefore the asymptotic complexity of our dynamic programming algorithm is $\mathcal{O}(nMS)$. Since the dynamic program is repeated for at most $\log MS$ iterations, our algorithm for LBFS has computational complexity $\mathcal{O}(nMS \log MS)$ and state space $\mathcal{O}(nMS)$. An interesting feature of the above dynamic program is that its variables are binary and therefore each state occupies only one bit of memory making it possible to implement very large size problems on microcomputer supported environments.

A similar dynamic program can be formulated for LAFS. However, the average relative gap of H_2 for the LAFS problem (as reported in Table 2.1) shows that such

an algorithm is not needed. Hence, we proceed in the next subsection by comparing the average performance of the designs discussed in this paper. The following comparisons will be made:

- look-behind flowshop versus traditional flowshop,
- look-behind flowshop versus flexible flowshop,
- look-ahead flowshop versus traditional flowshop,
- look-ahead flowshop versus flexible flowshop.

Computational Results

In this subsection we study the effects of important characteristics of the production environment (in particular, stage workloads and variation among processing times of the jobs at each stage) on four different configurations (we are restricting ourselves to the two-stage cases and at most two machines in each stage): look-behind flowshop (as defined in Section 2.1), look-ahead flowshop (as defined in Section 2.1), flowshop (as traditionally defined, with all jobs following the same routing of stages) and flexible flowshop (as defined in Section 2.1). For all environments job preemption is not allowed.

More precise definitions of the characteristics of the production environment we considered, are the following:

(a) Stage Workload Ratio (LR): it reflects the ratio of workloads at the two stages of the traditional flowshop environment, i.e.

$$LR = \frac{\sum_{i=1}^{n} p_{i1}}{\sum_{i=1}^{n} p_{i2}}$$

(b) Coefficient of Variation of processing times at stage k, CV_k $(k = 1, 2)$: it reflects the variability of processing times of jobs at a specific stage k $(k = 1, 2)$ (as a reminder,

the processing time of each job is deterministic, and the variability we are referring to is among processing times of jobs at a stage), i.e.

$$CV_k = \frac{\sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(p_{ik} - \bar{p_k})^2}}{\bar{p_k}}, \ where \ \bar{p_k} = \frac{\sum_{i=1}^{n} p_{ik}}{n}, \ k = 1, 2.$$

In Tables 2.2, 2.3 we report the results of our computational experiments. We generated random scheduling problems for a flowshop environment with 15 jobs within the following ranges of the workload ratio:

- $LR \in [0.6, 0.8]$ for unbalanced workloads with lower workload on stage 1,
- LR $\in [0.9, 1.1]$ for balanced workloads,
- $LR \in [1.2, 1.4]$ for unbalanced workloads with lower workload on stage 2,

and the following ranges of coefficients of variation:

- $CV_k \in [0.5, 0.7]$ for low processing time variability at stage k,
- $CV_k \in [0.9, 1.1]$ for medium processing time variability at stage k,
- $CV_k \in [1.2, 1.4]$ for high processing time variability at stage k.

There is a total of 27 combinations of ranges for the various parameters, and for each combination 5 random scheduling problems were generated. Thus a total of 135 test problems were solved for each table. For each combination of parameters, we report the average over the 5 test problems within that range of the following two quantities in Table 2.2:

i) percentage deviation of the makespan performance of the flowshop from the makespan performance of the look-behind flowshop as determined by the DP1 algorithm, i.e. $\frac{C_{FS} - C_{LBFS}}{C_{LBFS}} 100\%$, and

ii) percentage deviation of the makespan performance of the look-behind flowshop as determined by the DP1 algorithm, from the makespan performance of the flexible flowshop resulting from an application of an optimal algorithm for 2FFS (see Kouvelis and Vairaktarakis (1993)), i.e. $\frac{C_{2FFS}-C_{LBFS}}{C_{LBFS}}100\%$.

Therefore, a $-$ sign indicates that the 2FFS outperforms LBFS, while no sign indicates that LBFS outperforms 2FFS. Similarly, in Table 2.3 we report the following:

iii) $\frac{C_{FS}-C_{LAFS}}{C_{LAFS}}100\%$, where C_{LAFS} is the makespan determined by the H_2 heuristic,

iv) $\frac{C_{2FFS}-C_{LAFS}}{C_{LAFS}}100\%$.

As a result, a $-$ sign indicates that the 2FFS outperforms LAFS, while no sign indicates that LBFS outperforms 2FFS. As mentioned above, the figures reported in Tables 2.2 and 2.3, represent the average deviation over 5 random problems. The deviation of each of the 5 problems is very close to the reported average for all the combinations of load ratio and processing time variations. This observation is one of the reasons for experimenting with the particular combinations described above. Below, we summarize and interpret the results of Table 2.2. We start by comparing the performance of LBFS environments with traditional flowshops.

1a) The look-behind flowshop environment significantly outperforms traditional flowshops. The makespan performance improvements are in the range of 0-35%. The differences are negligible for $LR \in [0.6, 0.8]$, minor for $LR \in [0.9, 1.1]$ and pronounced for $LR \in [1.2, 1.4]$. Specifically, we have that the average percentage deviation in performance of the two environments is 2.61% for balanced workloads across the two stages, 0.4% for unbalanced workloads with lower workload on stage 1 and 29.8% for unbalanced workloads with lower workload on stage 2.

1b) For unbalanced workload flowshop environments where stage 2 is overloaded, the second stage acts as bottleneck in the same fashion for both LBFS and FS. Hence,

Table 2.2. Performance Comparison of LBFS with FS and 2FFS

Workload Ratio LR	Coefficient of Variation CV_1	Coefficient of Variation CV_2					
		$[0.5, 0.7]$		$[0.9, 1.1]$		$[1.5, 2]$	
		FS	2FFS	FS	2FFS	FS	2FFS
$[0.6, 0.8]$	$[0.5, 0.7]$	0.0	-17.2	0.0	-15.5	0.0	-14.0
	$[0.9, 1.1]$	0.0	-17.0	0.0	-19.4	0.0	-14.9
	$[1.5, 2]$	0.0	-16.2	0.0	-12.1	4.1	-13.1
$[0.9, 1.1]$	$[0.5, 0.7]$	1.1	-3.4	4.6	-2.9	2.2	-2.1
	$[0.9, 1.1]$	0.9	-4.9	4.5	-4.2	2.3	0.0
	$[1.5, 2]$	3.7	0.0	4.2	0.0	0.0	0.0
$[1.2, 1.4]$	$[0.5, 0.7]$	27.2	12.8	24.1	7.3	26.7	11.3
	$[0.9, 1.1]$	28.8	7.9	34.3	1.1	34.1	14.2
	$[1.5, 2]$	26.7	10.8	34.9	13.6	31.8	13.3

in this case the two environments have similar performance. When stage 1 is overloaded, however, the first stage becomes the bottleneck and the multistation line of LBFS forces (on the average) 29.8% less idle time on stage 2 as compared with the idle time forced on stage 2 of FS. Hence, irrespective of the variability in processing times the LBFS environment performs significantly better than FS.

1c) For balanced workloads between the two stages, the difference in the performance of LBFS and FS is minor especially in the presence of small or large CV_2 values (i.e. $CV_2 \in [0.6, 0.8] \bigcup [1.2, 1.4]$). When the variability CV_2 is small, then since $LR \in [0.9, 1.1]$, the idle time forced into stage 2 of FS is minor and FS performs nearly as well as LBFS. When CV_2 is large, then the Johnson's algorithm is able to find enough jobs with long stage 2 tasks so that the total idle time inserted into stage 2 of FS is about the same as the idle time inserted into stage 2 of LBFS. However, when $CV_2 \in [0.9, 1.1]$ such jobs are hard to find and as a result the performance difference of LBFS is more pronounced.

We now proceed to compare the performance of LBFS with the two machine flexible flowshop environment:

2a) The 2FFS environment performs better than the LBFS environment for balanced workloads as well as for unbalanced workloads with stage 2 overloaded. For balanced workloads across stages, the makespan performance improvements are in the range of 0-4.94% while for unbalanced workloads where stage 2 is overloaded the difference in performance is in the range of 12.12-19.48%. For unbalanced workloads where stage 1 is overloaded, the LBFS performs better than the 2FFS and the performance difference is in the range 1.1-14.2%. Specifically we have that the average percentage deviation in performance of the two environments is 1.96% for balanced workloads across the two stages, 15.51% for unbalanced workloads with lower workload on stage 1 and 10.25% for unbalanced workloads with lower workload on stage 2 (as a reminder, the workload definition relates to the two stages of a traditional flowshop).

2b) It is evident that when stage 1 is overloaded LBFS is superior to 2FFS while 2FFS outperforms LBFS when stage 2 is overloaded. Also, when 2FFS is superior, the performance benefits realized are greater than the benefits when LBFS is superior. This happens because when stage 2 is overloaded the stage 1 machines of LBFS are unable to alleviate the machine at stage 2 which have to stay idle. On the contrary, when stage 1 is overloaded, the total workload is divided between the two machines of 2FFS thus avoiding unnecessary idle time. This means that the processing time flexibility enables the 2FFS to perform relatively well even in the case where stage 1 is overloaded. This observation along with the fact that the 2FFS environment has one machine less than the LBFS show that the 2FFS environment is more robust under processing time uncertainty. However, if $LR \geq 1.2$ the LBFS is the design of choice.

2c) Table 2 indicates that when $LR \in [0.9, 1.1]$ the 2FFS design which consists of only two machines, outperforms both LBFS and LAFS designs that consist of three machines. This observation is contrary to the popular belief that adding machines

to the production system improves performance. It also indicates that for balanced workloads it is more profitable to invest in sophisticated tool magazines than to invest in new equipment.

Similar comments can be made for the performance of LAFS versus FS and 2FFS. Detailed results are reported in Table 2.3.

Table 2.3. Performance Comparison of LAFS with FS and 2FFS

Workload Ratio LR	Coefficient of Variation CV_1	Coefficient of Variation CV_2					
		$[0.5, 0.7]$		$[0.9, 1.1]$		$[1.5, 2]$	
		FS	2FFS	FS	2FFS	FS	2FFS
$[0.6, 0.8]$	$[0.5, 0.7]$	49.3	21.7	40.3	19.4	28.2	10.3
	$[0.9, 1.1]$	43.0	21.8	38.2	20.1	36.4	15.9
	$[1.5, 2]$	34.4	12.5	41.4	17.2	30.4	15.2
$[0.9, 1.1]$	$[0.5, 0.7]$	2.1	-3.5	5.3	-5.3	4.4	0.0
	$[0.9, 1.1]$	3.5	0.0	7.7	0.5	0.0	-2.1
	$[1.5, 2]$	0.0	-4.1	0.0	-4.0	0.0	-2.1
$[1.2, 1.4]$	$[0.5, 0.7]$	0.0	-12.0	0.0	-10.8	0.0	-14.0
	$[0.9, 1.1]$	2.3	-11.1	0.0	-14.2	0.0	-12.7
	$[1.5, 2]$	0.0	-10.5	0.0	-13.7	0.0	-12.0

Since LBFS (LAFS) is more appropriate when stage 1 (stage 2) is overloaded, the process of choice is immediately identified by knowing the value of LR. For this reason, we did not include the comparison of LBFS versus LAFS in our experiment. In the table presented in the next section we provide comparative results for the designs we considered and link our findings with the concept of *design for schedulability.*

2.5 Comparative Results

Making process choice decisions in environments with significant processing flexibility (mostly due to the presence of flexible machines) is a difficult task. In this paper we compared the throughput performance of four different process configurations for production systems with two flexible production stages (a production stage

might be a machine or a manufacturing cell). Our base design is the 2-machine traditional flowshop environment. Increased workloads on any of the stages trigger the redesigning of the production system, also known as *conversion process*. We recommend the following three design options as possible ways to operate such flowshop environments:

(a) A flexible flowshop environment, with the two flexible machines being tooled in a non-identical fashion. The processing flexibility across stages is exploited by allowing some of the jobs to be processed completely on only one of the machines while for others the production task is split between the two machines. As a result, the flexible flowshop assumes that each of its two machines is equipped with sophisticated tool magazines. The production planning task for the flexible flowshop is more involved than that of the traditional flowshop as the production tasks of each job have to be allocated to the two machines and then be appropriately scheduled. For workloads in the range $0.9 \leq LR \leq 1.1$, it is more important to invest in the tool magazine than to invest in an extra machine for the first or second stage.

(b) A flexible look-behind flowshop environment, with the two flexible machines at stage 1 being tooled in an identical fashion. This environment materializes important benefits over the traditional flowshop in the case where $LR \geq 1.2$ but it comes in the expense of more complicated routing control and production scheduling.

(c) A flexible look-ahead flowshop environment, with the two flexible machines at stage 2 being tooled in an identical fashion. Switching to this design is justified in case that $LR \leq 0.8$ and it requires more sophisticated routing control and production scheduling.

To keep our summary of the results of the paper at a brief level, we refer the reader to a table of comparative results for the four environments (see Table 2.4). These results, mostly supported by the computational study of Section 2.4, can be

used by the production manager as guidelines for making process choice decisions in flexible manufacturing environments.

Table 2.4. Comparative Results for Alternative Two-Stage Process Configurations: Flowshop (FS), Flexible Flowshop (2FFS), Look-Behind Flowshop (LBFS) and Look-Ahead Flowshop (LAFS)

Dimensions of Comparison	Process Configuration			
	Flowshop (FS)	Flexible Flowshop (2FFS)	Look-Behind Flexible Flowshop (LBFS)	Look-Ahead Flexible Flowshop (LAFS)
Number of machines	Two	Two	Three	Three
Mode of flexibility	Unidirectional Routing	Processing Flexibility	Routing flexibility in stage 1	Routing flexibility in stage 2
Sophistication of tool magazine	Low	High	Low	Low
Characteristics of Environment that favors its Throughput Performance	Balanced workloads and small or large processing time variations in each machine	Balanced workloads with $0.9 \leq LR \leq 1.1$	Unbalanced workloads with $LR \geq 1.2$	Unbalanced workloads with $LR \leq 0.8$

It is apparent from the above discussion that an important factor for process choice decisions is the schedulability of the chosen design. This factor is further analyzed in the next section.

2.6 Discussion

Modern industry is revolutionalized by the concept of *design for flexibility* analyzed in detail by Luggen (1991), Maleki (1991). However, the performance benefits realized by increased system flexibility come in the expense of significant capital investment, more involved routing control and more complicated production scheduling. It is unrealistic to believe that the issue of productivity is resolved instantaneously by super-flexible designs. In managing flexible designs, the issue of scheduling is a serious short term (hours-minutes) concern to the manager of the shop floor. Flexible designs that are difficult to schedule not only require a lot of time and effort to be managed but also incur logistic costs. For these reasons *a manager should balance the benefits of flexibility with the ease of schedulability.* This last statement is the essence of the concept: *design for schedulability.*

A nice analog of the *flexibility and schedulability* of production systems can be

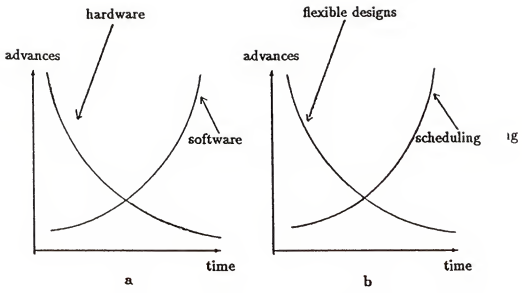


Figure 2.2. Analogies between Flexible Designs and Computer Systems

given by the *hardware and software* of computer systems. In a computer system the hardware aspect remains almost unchanged throughout the useful life of the system; just like the machines of a production system. Technological advances can usually improve the performance of a flexible design or a computer by a small factor. As is usually the case, however, emerging technologies provide advanced flexible designs and computers that render previous technologies significantly inferior.

On the other hand, the schedulability of a flexible design resembles the software aspect of computer systems. An algorithmic breakthrough for some problem provides solutions that are much faster and accurate than a breakthrough in computer technology. Similarly, new results in scheduling theory have greater impact towards improving the performance of a design, more so than any minor modification of the design can have. In addition, advances in scheduling theory are much more often than advances in the design of flexible systems, just like advances in software are much more often than advances in hardware.

The advances of hardware and software aspects of computer systems through time are usually graphed as in Figure 2.2a. The above discussion shows that a similar graph governs the technological advances of the design and scheduling aspects of production systems (see Figure 2.2b).

2.7 Conclusion

In this paper we address the scheduling problem in look-behind and look-ahead flexible flowshops which is magnified as compared with the traditional flowshop environment. Although the LBFS and LAFS environments treated in this paper possess significant flexibility, the theory developed in Section 2.3 resolves the issue of scheduling for these environments by finding fast heuristics that have negligible optimality gap. As a result the LBFS and LAFS designs become more attractive and more accessible to managers that have to choose their production system. We compare LBFS and LAFS with several other 2-stage flowshop designs in order to find the production characteristics that favor each design. Our experiments lead to a set of guidelines useful to a production manager who has to make process choice decisions. These guidelines hint that in choosing the right production environment, the schedulability of the chosen design weighs very heavily into decision making. This observation leads to the novel concept of *design for schedulability* which is discussed and given an analog from the hardware and software of computer systems.

CHAPTER 3

MINIMIZING MAKESPAN IN HYBRID FLOWSHOPS

3.1 Introduction

In this chapter we introduce a multiple stage flowshop environment with several machines in each stage. We refer to this design as the *hybrid flowshop* (*HFS*), see Gupta (1988). The HFS design has k stages each of which consists of m_i parallel identical machines, $i = 1, 2, \ldots, k$. Jobs flow through from stage to stage by utilizing any of the machines in the stage while satisfying the flowshop constraints (see Figure 3.1). Evidently, the HFS environment possesses significant routing flexibility within stages.

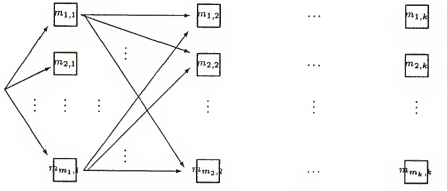


Figure 3.1. The hybrid flowshop environment

The design just described is frequently encountered in flexible manufacturing systems (FMS) (see Maleki (1991) for detailed description of FMS's), where each production stage might be either a flexible machine or a flexible manufacturing cell. Flexible machines are equipped with tool magazines that can accommodate up to 160 different tools. In case that the tool magazine of all the machines within a stage can carry a

great number of tools, the first operation of a job can be processed by either of the machines. Then, the production system enjoys flexibility within a stage.

Flexible manufacturing cells (see Luggen (1991)) consist of a set of flexible machines, able to perform a variety of operations. Each of the manufacturing cells is a multistation line. For an environment with k flexible manufacturing cells, the HFS problem amounts to allocating the set of jobs to the various lines of the various cells, in such a way that the total idle time incurred by the last cell in the production line, is minimized. Such a requirement optimizes the throughput performance as well as the machine utilization of the production system.

The study of scheduling problems in flexible manufacturing systems has attracted significant attention in recent years including Afentakis (1986), Erschler *et al.* (1985), Ghosh and Gaimon (1992), Hermann and Lee (1992), Kouvelis and Vairaktarakis (1993), Langston (1987), Lee and Hermann (1993), Lee and Vairaktarakis (1993), Shanker and Tzen (1985), Shriskandarajah and Ladet (1986), Shriskandarajah and Sethi (1989), Stecke (1985,1992) and Wittrock (1988), due to the importance of such systems for small-to-medium batch manufacturing.

The layout of this chapter is as follows. A formal definition of the problem is given in Section 3.2. A fast heuristic algorithm H for the two stage hybrid flowshop problem and minimum makespan objective is developed in Section 3.3.1. We show that the worst case error bound of H is relatively small and that its average performance is very good. In Section 3.3.2 we extend H to the hybrid flowshop with more than two stages and show that the resulting algorithm H' extends the current best error bound for the traditional k machine flowshop problem, to the general hybrid flowshop problem. Closing remarks are made in Section 3.4.

3.2 Problem Description and Literature Review

Consider a k-stage flowshop environment with m_i parallel identical machines in each stage, $i = 1, 2, \ldots, k$. We refer to this production environment as the hybrid flowshop (*HFS*) and we denote it as $FS_{m_1,m_2,\ldots,m_k}$. A set J of n jobs is processed on the above environment. Each job $J_i \in J$ consists of k tasks; namely $(i,1),(i,2),\ldots,(i,k)$ with corresponding processing time requirements $p_{i1}, p_{i2}, \ldots, p_{ik}$. The first stage task of J_i can be processed by either of the m_1 first stage machines. Upon completion of the $(i,1)$ task, the $(i,2)$ task can commence on any of the m_2 stage 2 machines, and so on (see Figure 3.1). No machine can process more than one task at a time and no preemption is allowed for the jobs. Our objective is to minimize makespan.

The Hybrid flowshop or its subsystems are very common production environments of many industrial sectors that include semiconductor manufacturing (see Herrmann and Lee (1992), Lee and Herrmann (1993)), glass container industry (see Paul (1979)), cable manufacturing (see Narasimhan and Panwalker (1984)) and others. The HFS design is a generalization of the m parallel identical machine environment (mP) where it is assumed that $p_{i2} = \ldots = p_{ik} = 0$ for every $J_i \in J$. The mP environment has been well studied over the last 25 years by several researchers. A review including most results in this area is given by Cheng and Sin (1990). Garey and Johnson (1979) have shown that minimizing makespan in the mP environment is $\mathcal{NP}$-complete.

Several special cases of the 2-stage hybrid flowshop problem have been considered in the literature. Arthanary and Ramaswamy (1971) developed a branch and bound algorithm for $FS_{m,1}$ which is inefficient for problem sizes of more than 10 jobs. Several heuristic algorithms without worst case error bound analysis have been proposed for $FS_{m,1}$ and $FS_{1,m}$ by Gupta (1988), Gupta and Tunc (1991), Rao (1970) and others. These heuristics are evaluated by establishing a lower bound and then computing the average relative gap of the heuristic solution from the lower bound. However,

the lower bounds that have been proposed for $FS_{m,1}$ and $FS_{1,m}$ are either trivial or loose.

Heuristics accompanied with worst case error bound analysis are also found in literature. We tabulate these results in Table 3.1. For each case we report the complexity, the worst case error bound, the sequence used by the heuristic as well as the appropriate reference.

Table 3.1. Previous literature on the 2-stage HFS problem

2-Stage Hybrid Flowshop Literature				
Design	Complexity	Error bound	Sequence used	Reference*
$1 \times m$	$\mathcal{O}(n)$	$3 - \frac{1}{m}$	arbitrary	[57]
$1 \times m$	$\mathcal{O}(n \log n)$	$3 - \frac{3}{m} + \frac{1}{m^2}$	Johnson	[57]
$1 \times m$	$\mathcal{O}(n \log n)$	$2 - \frac{1}{m}$	Johnson(1)	[43]
$m \times 1$	$\mathcal{O}(n \log n)$	$2 - \frac{1}{m}$	Johnson(2)	[43]
$m \times m$	$\mathcal{O}(mn \log n)$	$3 - \frac{1}{m}$	Johnson	[57]
$m \times m$	$\mathcal{O}(mn \log n)$	$\frac{7}{3} - \frac{2}{3m} \leq r \leq 3 - \frac{1}{m}$	unspecified	[57]
$m \times m$	$\mathcal{O}(n)$	$3 - \frac{1}{m}$	arbitrary	[40]
$m \times m$	$\mathcal{O}(n \log n)$	2	MJO(3)	[11]
$m \times m$	$\mathcal{O}(n \log n)$	$\frac{5}{2}$	$SORTA$(4)	[40]
$m \times m$	$\mathcal{O}(n \log n)$	2	$SORTB$(5)	[40]

* see reference section

1. Johnson with respect to $(p_{i1}, \frac{1}{m} p_{i2})$
2. Johnson with respect to $(\frac{1}{m} p_{i1}, p_{i2})$
3. MJO is the Modified Johnson's order; see Buten and Shen (1973)
4. SORTA is the shortest processing time order with respect to p_{i1}
5. SORTB is the shortest processing time order with respect to p_{i2}

Since the mP environment is a special case of the HFS, minimizing makespan in this environment is $\mathcal{NP}$-complete as well. Also, the traditional k-machine flowshop problem (see Baker (1974)), which is a special case of the hybrid flowshop, is $\mathcal{NP}$-complete in the strong sense even for $k = 3$ (see Garey and Johnson (1979)). These observations direct our attention to heuristic algorithms. This is the focus of the next section.

3.3 Heuristic Algorithms for the Hybrid Flowshop

3.3.1 Two Stage Production

In this subsection, we focus our discussion to the FS_{m_1,m_2} design. The theory developed will be extended to $FS_{m_1,m_2,\ldots,m_k}$ design in the next subsection. Since FS_{m_1,m_2} is a hybrid between the multiple parallel identical machine environment mP and the traditional two machine flowshop $FS_{1,1}$, we are motivated to develop heuristic algorithms that are hybrids of existing good algorithms for these two environments. Minimizing the makespan in $FS_{1,1}$ is solved in $\mathcal{O}(n \log n)$ time by Johnson's algorithm which we refer to as *JA* (see Johnson (1954)). The JA algorithm follows immediately from the rule: *Job* j_i *precedes job* j_j *if* $\min\{p_{i1}, p_{j2}\} \leq \min\{p_{j1}, p_{i2}\}$.

Most successful heuristics for the mP environment utilize an ordering S of the jobs along with the *first available machine* rule (*FAM*). According to the first available machine rule, the job to be scheduled next in the mP environment, is assigned to the first machine that becomes available i.e. the machine that finishes first the job (if any) previously assigned to it. Depending on the order S, the above heuristic produces different solutions.

In the following we develop a mirror image of the FAM rule and call it *last busy machine* rule (*LBM*). This rule will be used later to assign jobs to the second stage machines. Given a constant $T > 0$ and an ordering S of jobs where each job j has a specified processing time p_j, a description of the rule is given next.

LBM rule:

Step 1. Set $t_m := T$ for $m = 1, \ldots, m_2$.

Step 2. Let j be the last unscheduled job of S and m a machine with largest t_m. Schedule job j on machine m to finish at time t_m.

Step 3. Set $t_m := t_m - p_j$ and $S := S - \{j\}$. If $S \neq \emptyset$ then goto Step 2 else Stop.

Remark: The value of t_m is the time that machine m becomes busy. In Step 2 we assign the job j to the machine with largest t_m, i.e. the last busy machine. Hence we call this rule the last busy machine rule. Also, note that the value of T is only a reference point and has no effect on the allocation of jobs to machines. With this background we can present the following heuristic which uses the Johnson's order in conjunction with the FAM and LBM rules.

Heuristic H:

Step 1. Apply JA with respect to the processing times $\{(\frac{1}{m_1}p_{i1}, \frac{1}{m_2}p_{i2}) : i = 1, 2, \ldots, n\}$. Let S be the resulting sequence.

Step 2. Apply the FAM rule on the stage 1 tasks of the sequence S

Step 3. Apply the LBM rule on the stage 2 tasks of the sequence S

Step 4. On each stage 2 machine m, reorder the tasks assigned to m (during Step 3) according to nonincreasing completion times of the corresponding stage 1 tasks. Let S_m be the resulting order for $m = 1, 2, \ldots, m_2$.

Step 5. On each stage 2 machine m, schedule the tasks in S_m in this order, as soon as possible

At Step 1 of H a sequence S of jobs is produced, at Step 2 an assignment of $(i, 1)$ tasks on the first stage machines is made and at Steps 3,4 and 5 the tasks of stage 2 are scheduled. In particular, Step 3 determines which tasks will be processed by each stage 2 machine, Step 4 determines the order of stage 2 tasks within a stage 2 machine and Step 5 proceeds with the scheduling of the stage 2 tasks on stage 2 machines. Since JA requires $\mathcal{O}(n \log n)$ time, this is also the computational effort required by H.

Example: Consider the case $m_1 = m_2 = 2$ and a set of 5 jobs with processing time

requirements (1,3), (1,4), (2,4), (3,6) and (6,2). The jobs are already written according to Johnson's order. The Gantt chart of an application of H on this example is given in Figure 3.2.

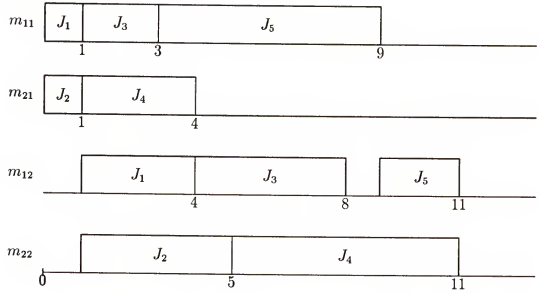


Figure 3.2. The Gantt chart of an application of H

The next lemma will be used in finding the error bound of H. For this lemma we introduce the following *auxiliary* problem. Replace the first stage machines $m_{11}, m_{21}, \ldots, m_{m_1,1}$ by a dummy machine M_1 and replace the processing time requirement of p_{i1} units on one of $m_{11}, m_{21}, \ldots, m_{m_1,1}$ by the requirement of $\frac{1}{m_1}p_{i1}$ units on M_1. Similarly, replace the second stage machines $m_{12}, m_{22}, \ldots, m_{m_2,2}$ by a dummy machine M_2 and replace the processing time requirement of p_{i2} units on one of $m_{1,2}, m_{2,2}, \ldots, m_{m_2,2}$ by the requirement of $\frac{1}{m_2}p_{i2}$ units on M_2. This way we define an auxiliary 2-machine flowshop problem on M_1, M_2. We will refer to this problem as AFS.

Lemma 2 *Let C_{LB} be the completion time found by JA for the auxiliary problem. Then, $C_{LB} \leq C_{FS_{m_1,m_2}}$.*

Proof. Consider an optimal solution S^* for FS_{m_1,m_2}. Let $S = \{J_1, J_2, \ldots, J_n\}$ be the set of jobs ordered in nondecreasing order of completion times of the stage 1 tasks in

S^*. Consider the partial schedule of S^* consisting of the first i stage 1 tasks of S and the last $n-i+1$ stage 2 tasks of S. Since in S^* the last $n-i+1$ stage 2 tasks of S start no earlier than the completion time of the first i stage 1 tasks of S (due to the flowshop constraints), we have that

$$C_{FS_{m_1,m_2}} \geq \frac{1}{m_1}\sum_{j=1}^{i} p_{j1} + \frac{1}{m_2}\sum_{j=i}^{n} p_{j2} \quad for \quad every \quad 1 \leq i \leq n.$$

For the AFS problem (where the processing time requirements for J_i are $(\frac{1}{m_1}p_{i1}, \frac{1}{m_2}p_{i2})$), schedule the jobs according to sequence S. Let C_S be the resulting makespan. Then, it is clear that

$$C_S = \frac{1}{m_1}\sum_{j=1}^{i_0} p_{j1} + \frac{1}{m_2}\sum_{j=i_0}^{n} p_{j2},$$

where J_{i_0} is the last job whose stage 2 task starts immediately after the completion of the corresponding stage 1 task (note that such a task always exists; since J_1 satisfies this property).

Combining the last two expressions we get that $C_S \leq C_{FS_{m_1,m_2}}$. However, the sequence S is not necessarily optimal for the AFS problem and hence $C_{LB} \leq C_S$. The last two relations establish that $C_{LB} \leq C_{FS_{m_1,m_2}}$. This completes the proof of the lemma. □

Remark: C_{LB} is the makespan found by applying JA to the AFS problem. Lemma 2 shows that C_{LB} is also a lower bound for the FS_{m_1,m_2} problem. This is the reason for using the notation C_{LB} rather than C_{AFS}.

The following theorem establishes the worst case performance of H. Let C_H be the makespan value obtained by H.

Theorem 6 $\frac{C_H}{C_{FS_{m_1,m_2}}} \leq 2 - \frac{1}{m}$, *where* $m = \max\{m_1, m_2\}$, *and the bound is tight.*

Proof. Let $S = \{J_1, J_2, \ldots, J_n\}$ be the order obtained at Step 1 of H. After the scheduling of stage 1 tasks at Step 2, the completion time of the first task of J_i, say

C_i, is

$$C_i \leq \frac{1}{m_1}\sum_{j=1}^{i-1} p_{j1} + p_{i1} = \frac{1}{m_1}\sum_{j=1}^{i} p_{j1} + \frac{m_1 - 1}{m_1} p_{i1}$$

for every $1 \leq i \leq n$, because the FAM rule is used for stage 1 tasks.

At Step 3 of H, suppose that the LBM rule is applied on the stage 2 tasks with respect to $T := (2 - \frac{1}{m})C_{FS_{m_1,m_2}}$. Let s_i be the starting time of the stage 2 task of J_i upon completion of Step 3. We will show that $C_i \leq s_i$ for all i. This will imply that if we assign jobs to stage 1 machines by the FAM rule and jobs to stage 2 machines by the LBM rule and then shift all the jobs in the second stage machines to the left as much as possible without violating the flowshop constraints, then the makespan of the schedule will not exceed $(2 - \frac{1}{m})C_{FS_{m_1,m_2}}$.

To see that $s_i \geq C_i$ note that for every $1 \leq i \leq n$

$$s_i \geq T - \frac{1}{m_2}\sum_{j=i}^{n} p_{j2} - \frac{m_2 - 1}{m_2} p_{i2}$$

for every $1 \leq i \leq n$, because the LBM rule is used for stage 2 tasks. Then,

$$s_i - C_i \geq (2 - \frac{1}{m})C_{FS_{m_1,m_2}} - \frac{1}{m_2}\sum_{j=i}^{n} p_{j2} - \frac{m_2 - 1}{m_2} p_{i2} - \frac{1}{m_1}\sum_{j=1}^{i} p_{j1} - \frac{m_1 - 1}{m_1} p_{i1}.$$

Note that

$$\frac{1}{m_1}\sum_{j=1}^{i} p_{j1} + \frac{1}{m_2}\sum_{j=i}^{n} p_{j2} \leq C_{LB} \leq C_{FS_{m_1,m_2}} \quad for \quad every \quad 1 \leq i \leq n$$

as we showed in the proof of Lemma 2. Also,

$$\frac{m_1 - 1}{m_1} p_{i1} + \frac{m_2 - 1}{m_2} p_{i2} \leq \frac{m - 1}{m}(p_{i1} + p_{i2}) \quad where \quad m = \max\{m_1, m_2\}$$

$$\leq \frac{m - 1}{m} C_{FS_{m_1,m_2}}.$$

Hence,

$$s_i - C_i \geq T - (1 + \frac{m - 1}{m})C_{FS_{m_1,m_2}} = 0.$$

Therefore, the schedule produced by concatenating, as described above, the partial schedules of Step 2 and Step 3 has a makespan no greater than $(2 - \frac{1}{m})C_{FS_{m_1,m_2}}$. It

is easy to check that Steps 4 and 5 can only improve the makespan performance of the above schedule. This completes the upper bound performance of H.

To see that the bound of $2-\frac{1}{m}$ is tight for H, consider the case where $p_{j2}=0$ for all jobs. Then the heuristic H reduces to the RDM list scheduling heuristic studied by Graham (1966) which has a tight worst case error bound of $2-\frac{1}{m_1}$. In a problem where $p_{j2}=0$ for all jobs and $m_1 \geq m_2$ the error bound of $2-\frac{1}{m}$ is tight for the heuristic H. This completes the proof of the theorem. □

We would like to compute the average deviation of H from the optimal solution of FS_{m_1,m_2}. In light of Lemma 2, we perform a computational experiment where we compute the average relative gap of H, from the lower bound C_{LB}. This can be done very efficiently since both of C_{LB}, C_H can be computed in polynomial time. In order to get tight relative gaps we will develop four more lower bounds which along with C_{LB} are going to be used for our comparisons.

In the optimal solution S^* for FS_{m_1,m_2}, there must be idle time on the stage 2 machines due to the flowshop constraints of the first job of each of the stage 1 machines. We distinguish the following two cases:

Case i: $m_1 \geq m_2$

Suppose that $J^1, J^2, \ldots, J^n$ is the order of jobs in nondecreasing order of stage 1 processing times, with corresponding processing times $p_{J^1,1}, \ldots, p_{J^n,1}$. It is clear that because of the flowshop constraints there will be at stage 2 a machine with idle time no less than $p_{J^1,1}$, a machine with idle time no less than $p_{J^2,1}$, ..., a machine with idle time no less than $p_{J^{m_2},1}$. As a result, the optimal makespan cannot be less than the average workload plus the necessary idle time of stage 2, i.e.

$$C_{FS_{m_1,m_2}} \geq \frac{\sum_{k=1}^{m_2} p_{J^k,1} + \sum_{i=1}^{n} p_{i2}}{m_2}.$$

Case ii: $m_1 < m_2$

Similarly, in this case there must be at stage 2 a machine with idle time no less

than $p_{J^1,1}, \ldots$, a machine with idle time no less than $p_{J^{m_1},1}$. The remaining $m_2 - m_1$ stage 2 machines cannot become operating until at least two tasks have been processed at stage 1. Since the shortest m_1 stage 1 tasks are already considered, there must exist at stage 2 a machine with idle time no less than $p_{J^{m_1+1},1} + p_{J^1,1}, \ldots$, a machine with idle time no less than $p_{J^{m_2},1} + p_{J^1,1}$. Therefore,

$$C_{FS_{m_1,m_2}} \geq \frac{\sum_{k=1}^{m_2} p_{J^k,1} + (m_2 - m_1)p_{J^1,1} + \sum_{i=1}^{n} p_{i2}}{m_2}.$$

Let us define $P_{k,s}$ to be the summation of the k shortest stage s tasks, $s = 1, 2$.

Then,

$$LB_1 = \frac{P_{m_2,1} + P_{n,2}}{m_2}$$

is a lower bound for $C_{FS_{m_1,m_2}}$ when $m_1 \geq m_2$, and

$$LB_2 = \frac{P_{m_2,1} + (m_2 - m_1)P_{1,1} + P_{n,2}}{m_2}$$

is a lower bound for $C_{FS_{m_1,m_2}}$ when $m_1 < m_2$.

To improve these bounds we can use the symmetry of the two-stage hybrid flowshop. Interchange the roles of stage 1 and stage 2 and let the tasks p_{i2}, p_{i1} play the role of stage 1 and stage 2 processing times respectively. Then, the makespan associated with any sequence S for the two stage hybrid flowshop problem, equals the makespan associated with the reverse sequence of S for the symmetric of the two stage hybrid flowshop problem. Then, we can derive the symmetric lower bounds

$$LB_3 = \frac{P_{m_1,1} + (m_1 - m_2)P_{1,2} + P_{n,1}}{m_1} \quad if \quad m_1 \geq m_2,$$

$$LB_4 = \frac{P_{m_1,2} + P_{n,1}}{m_1} \quad if \quad m_1 < m_2.$$

Summarizing our results we have the following lower bounds:

$$LB = \begin{cases} \max\{LB_1, LB_3, C_{LB}\} & \text{if } m_1 \geq m_2; \\ \max\{LB_2, LB_4, C_{LB}\} & \text{if } m_1 < m_2 \end{cases}$$

It is not difficult to see that none of $C_{LB}, LB_1, LB_2, LB_3, LB_4$ dominates the other. However, the previous best known lower bounds of $P_{1,1} + \frac{P_{n,2}}{m_2}$ and $P_{1,2} + \frac{P_{n,1}}{m_1}$ (see Gupta, 1988) for the two stage hybrid flowshop problem are dominated by LB_1, LB_3 respectively (and LB_2, LB_4 respectively).

In Table 3.2 we report the relative gap $\frac{C_H - C_{LB}}{C_{LB}} 100\%$. We report the average and worst case relative gap for different problem sizes and processing time ratios. We consider the problem sizes 2×4, 4×4 and 4×2, with 30, 40 and 50 jobs. The processing time ratios reported are $2:4$, $4:4$, $4:2$. For example, a ratio $2:4$ means that the processing time of $(i,1)$ is chosen randomly from a uniform distribution on $[1,20]$ and the processing time of $(i,2)$ is chosen randomly from a uniform distribution on $[1,40]$. This results to 27 ratio/size combinations. For each combination we report the average relative gap over 50 randomly selected problems. Thus, we solved a total of 1350 problems.

Table 3.2. Relative gap of H from the lower bound C_{LB}.

	$\frac{C_H - C_{LB}}{C_{LB}} 100\%$								
$m_1 \times m_2$	2×4	4×4	4×2	2×4	4×4	4×2	2×4	4×4	4×2
	$\sum_i p_{i1} : \sum_i p_{i2}$								
size	$2:4$			$4:4$			$4:2$		
30	1.8	1.6	0.9	1.9	2.9	0.5	1.6	3.5	2.6
40	1.2	1.0	0.5	1.3	2.6	0.4	1.4	2.9	2.2
50	0.8	0.4	0.2	0.9	2.3	0.3	1.1	2.7	1.7
Average	1.2	1.6	0.5	1.3	2.6	0.4	1.3	3.0	2.2

Table 3.2 suggests that the average relative gap for the $FS_{2,4}$ design is very small (1.2% on the average). Our heuristic exhibits similar performance for all 3 values of the workload ratio that we tested. For the $FS_{4,2}$ design the heuristic H exhibits an average relative gap of 2.2% for the workload ratio 4:2 where stage 2 is underloaded, while the corresponding gap for the ratios 2:4 and 4:4 is negligible. For the $FS_{4,4}$ design the values of the average relative gap are slightly bigger (2.4% on the average).

The average relative gap over all 27 combinations is 1.6%.

The above observations suggest that scheduling in a hybrid flowshop becomes more difficult as the total number of machines increases. Note that when the workload of stage 1 is at least as big as that of stage 2, the average relative gaps tend to assume greater values which reflects suboptimal stage 1 scheduling on the part of the heuristic. This can be explained from the fact that significant suboptimality due to stage 1 scheduling can be magnified by suboptimality of the scheduling of stage 2 tasks. Also, the average relative gap decreases as the number of jobs increase.

3.3.2 A Heuristic for the Hybrid Flowshop

The hybrid flowshop is a generalization over the parallel identical machine environment and the flowshop environment. These characteristics are captured by the heuristic H since for $m_1 = m_2 = 1$ our heuristic reduces to Johnson's algorithm and $\frac{C_H}{C_{FS_{1,1}}} = 1$. Also, if the hybrid flowshop consists of one stage only, then it reduces to the parallel identical machine environment and H reduces to the list scheduling heuristic for mP proposed by Graham (1966). In what follows we describe a heuristic algorithm H' for the general $FS_{m_1,m_2,\ldots,m_k}$ problem. The heuristic H' utilizes H and assumes that the number k of stages is even (otherwise we can introduce a dummy stage with zero machines).

Heuristic H':

1. Apply H on stages $2r-1, 2r$ and let S_r be the resulting schedule, for $r = 1, 2, \ldots, \frac{k}{2}$
2. Concatenate the schedules S_r, for $r = 1, 2, \ldots, \frac{k}{2}$. Eliminate all idle time between stage 1 tasks. Eliminate unnecessary idle time between stage 2 tasks.

The complexity of H' is $\mathcal{O}(kn \log n)$. The next theorem finds a bound to the error of H'.

Theorem 7 $\frac{C_{H'}}{C_{FS_{m_1,m_2,\ldots,m_k}}} \leq k - \frac{1}{\max\{m_1,m_2\}} - \frac{1}{\max\{m_3,m_4\}} - \ldots - \frac{1}{\max\{m_{k-1},m_k\}}$

Proof. Let C_r^* be the optimal makespan for stages $2r-1, 2r$. Then, by Theorem 6 we have that

$$C_{H'} \leq (2 - \frac{1}{\max\{m_1,m_2\}})C_1^* + (2 - \frac{1}{\max\{m_3,m_4\}})C_2^* + \ldots + (2 - \frac{1}{\max\{m_{k-1},m_k\}})C_{\frac{k}{2}}^*.$$

Clearly, $C_r^* \leq C_{FS_{m_1,m_2,\ldots,m_k}}$ for $r = 1, 2, \ldots, \frac{k}{2}$. Then,

$$\frac{C_{H'}}{C_{FS_{m_1,m_2,\ldots,m_k}}} \leq k - \frac{1}{\max\{m_1,m_2\}} - \frac{1}{\max\{m_3,m_4\}} - \ldots - \frac{1}{\max\{m_{k-1},m_k\}}$$

This completes the proof of the theorem. □

The above theorem is linked with existing theory for the flowshop problem as follows. If $m_1 = m_2 = \ldots = m_k = 1$ then the hybrid flowshop reduces to the traditional k-machine flowshop. In this case,

$$\frac{C_{H'}}{C_{FS_{1,1,\ldots,1}}} \leq \lceil \frac{k}{2} \rceil.$$

The latter bound coincides with the bound of Gonzalez and Sahni (1978), which to the best of our knowledge, is the best existing error bound for the k-machine flowshop problem.

Several cases where the error bound of H' is tight are known. The tightness of the k-machine flowshop for $k = 3, 4$ is given in Gonzalez and Sahni (1978). In case that the hybrid flowshop consists of only two stages, the tight examples given in the previous section provide tight examples for the hybrid flowshop as well. If the hybrid flowshop consists of one stage only, then H' reduces to RDM which as we mentioned earlier has a tight error bound of $2 - \frac{1}{m_1}$.

3.4 Conclusion

In this research we considered the problem of minimizing makespan in a hybrid flowshop. First we considered the case of two stage production for which we developed

a heuristic H which significantly improves all known error bounds for special cases of the problem and also extends the current best error bound for the designs $FS_{m,1}$ and $FS_{1,m}$ which we presented in Chapter 2. Based on the heuristic H we developed a heuristic H' for the general problem with more than two stages. We performed error bound analysis on H' which shows that the current best error bound for the k-stage traditional flowshop is attained by H', thus relating the results of this chapter with existing results on related problems. Our future research will focus on the implications of the routing control structure between adjacent stages to the makespan performance of the hybrid flowshop.

CHAPTER 4

THE TWO-STAGE FLEXIBLE FLOWSHOP ENVIRONMENT

4.1 Introduction

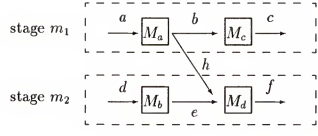
This chapter addresses a scheduling problem for a two stage production environment with significant processing flexibility across production stages. We refer to this environment as a two stage 'flexible' flowshop; for convenience in our further discussion we use the abbreviated term 2FFS. A job can be either completely processed at one of the stages or its processing is split in a prespecified fashion between the two stages. For the latter case the routing is in only one direction, from the 'upstream' stage to the 'downstream' stage, which preserves the routing structure of a traditionally defined flowshop (see Baker (1974)). We are assuming the presence of adequate storage space between the two stages.

The flexible flowshop environment just described is frequently encountered in flexible manufacturing systems (FMS) (see Maleki (1991) for detailed description of FMS's), where each production stage might be either a flexible machine or a flexible manufacturing cell. Flexible machines are designed with large tool magazines that can accommodate a great variety of tools (up to 120 different tools). Depending on the tool assignment, the flexible machine can perform a number of different tasks and in many cases can complete all the required operations of a job. The processing flexibility of such machines is accurately depicted in our flexible flowshop description as the operations of a job could be performed by either of the two machines or the operations of the job could be split between the two machines. Flexible manufacturing cells (see Luggen (1991)) consist of a set of flexible machines, with not necessarily all

of them being identically tooled, able to perform a variety of operations. For an environment with two flexible manufacturing cells, our flexible flowshop problem amounts to deciding on how to allocate a set of jobs between the two cells, with the additional option of some jobs allowed to be split between the two cells while preserving a fixed routing from the upstream cell to the downstream one.

The problem can be also motivated for environments with dedicated parallel lines allowed to share resources for the processing of some jobs. Such an environment is described in Figure 4.1.

Figure 4.1. A Flexible Flowshop Implementation in a Parallel Line Environment



Each one of the production stages is a multistation line. A job can be either processed completely on one of the lines or the first few processing steps are performed on line m_1 (the upstream line) and the remaining steps are processed on line m_2 (the downstream line). Such flexible flowshop examples can be found in electronic assembly line environments, where the flexibility of the automatic component placement machines allows the manager to operate the lines in a dedicated or shared resources mode.

The study of scheduling problems in flexible manufacturing and assembly systems has attracted significant attention in recent years (see Afentakis (1986), Erschler *et al.* (1985), Hermann and Lee (1992), Lee *et al.* (1993), Lee and Hermann (1993), Shanker and Tzen (1985), Shriskandarajah and Ladet (1986), Shriskandarajah and Sethi (1989), Stecke (1985), Wittrock (1988), Stecke (1992) and Ghosh and Gaimon

(1992)) due to the importance of such systems for small-to-medium batch manufacturing. Part of the above research literature evaluates the performance of algorithms which schedule parts through a manufacturing system referred to also as a 'flexible flowshop' (see Shriskandarajah and Ladet (1986), Shriskandarajah and Sethi (1989) and Wittrock (1988)). We would like to clearly point out the different use of the term in this chapter with that in the above literature. The previously described in the literature 'flexible flowshop' environments are multistage production environments with multiple identical parallel machines at each stage. The flexibility of these environments is derived from the ability to process a job on anyone of the parallel machines at a stage (processing flexibility within a production stage), while the flexibility of our described environment is a consequence of the fact that the job, or a part of it, could be processed on anyone of the production stages (processing flexibility across production stages). The common characteristic of all of the environments referred to as flexible flowshops is the preservation of the strict routing structure of the traditional flowshop environment, with parts being routed from an 'upstream' stage to a 'downstream' stage.

The class of flexible flowshops with processing flexibility across production stages, to the best of our knowledge, has not been studied in the research literature before. In this chapter we concentrate on the scheduling problem of the two stage flexible flowshop with makespan minimization objective. A formal definition of the problem and properties of optimal 2FFS schedules are provided in Section 4.2. We present a pseudopolynomial dynamic programming algorithm for the solution of the problem in Section 4.3.

The two stage flexible flowshop environment, as defined in this chapter, can be thought as a generalization of the two machine traditional flowshop and the two identical parallel machine environment (see further discussion on it in Section 4.2.1).

Motivated by this observation, we proceed to use algorithms developed for the solution of flowshop or parallel machine scheduling problems as heuristics for the 2FFS environment. In Section 4.4.1 we modify a parallel machine heuristic to better depict the characteristics of the 2FFS scheduling problem and prove that the resulting heuristic has a worst case error bound of 1.5. In Section 4.4.2 we analyze the performance of Johnson's algorithm (Johnson (1954)) when applied as a heuristic for the 2FFS scheduling problem and prove that its worst case error bound is 2. In Section 4.5 we perform a computational study of the effects of important characteristics of the production environment (in particular, stage workloads and variation among processing times of the jobs at each stage) on the performance of the 2FFS, two machine flowshop and two parallel machine environments. This study allows us to state in Section 4.6 managerial guidelines for the selection of the appropriate production process among the three that we analyzed based on the job characteristics processed by the production system.

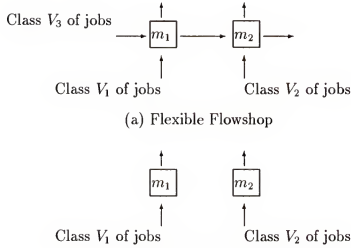
4.2 The Two Stage Flexible Flowshop Problem

4.2.1 Problem Description

A set of jobs $J = \{j_1, j_2, \ldots, j_n\}$ is given and every job j_i consists of two tasks, namely $(i,1)$ and $(i,2)$, with processing time requirements p_{i1} and p_{i2} respectively. All jobs are assumed to be available at time zero and no preemption is allowed for the tasks. The jobs in J are to be processed in a two stage flowshop which has the following type of processing flexibility. Every job j_i can be processed entirely on the upstream stage m_1 for $p_i = p_{i1} + p_{i2}$ periods or entirely in the downstream stage m_2 for p_i periods or task $(i,1)$ is processed on m_1 and task $(i,2)$ is processed on m_2. For maximizing the throughput, as well as the machine utilization of the system, we are interested in sequencing the jobs in J so that the resulting schedule on the two stage flexible flowshop minimizes the makespan.

It is of interest to note that the above described flexible flowshop represents a generalization of the traditional flowshop and the identical parallel machine shop. The flexible flowshop is an environment with three job classes; class V_1 of jobs require

Figure 4.2. The Flexible Flowshop as a Generalization of the Traditional Flowshop and Parallel Machine Environments



processing only at the upstream stage, class V_2 of jobs require processing only at the downstream stage and class V_3 of jobs require processing at both stages, (see Figure (4.2a)). The three classes of jobs, can be also recognized at the flexible flowshop environment described in Figure 1: class V_1 of jobs follow the route $a \rightarrow b \rightarrow c$, class V_2 of jobs follow the route $d \rightarrow e \rightarrow f$ and class V_3 of jobs follow the route $a \rightarrow h \rightarrow f$. It is now easy to observe that the parallel machine environment is an environment with only classes V_1 and V_2 of jobs (see Figure (4.2b)) and the traditional flowshop can be thought of as an environment with only one class of jobs that require processing both at the upstream and the downstream stage (see Figure (4.2c)).

Observe that in the case that $p_{i2} = 0$ for all $j_i \in J$, our flexible flowshop problem reduces to the problem of minimizing makespan on two identical parallel machines. This parallel machine scheduling problem is known to be ordinary $\mathcal{NP}$-complete (see Garey and Johnson (1979)), and therefore our problem, since it contains the above problem as a special case, is $\mathcal{NP}$-complete as well.

4.2.2 Properties of Optimal 2FFS Schedules

In this subsection we identify a number of optimal properties for 2FFS schedules. All these properties follow by simple interchange arguments that can be easily inferred from optimal properties of traditional two stage flowshop and job shop environments as documented in scheduling textbooks (Baker (1974)). First, given a solution S for the two stage 2FFS we partition the set of jobs J into the same three classes as in Figure (4.2a):

V_1: The jobs in J whose both tasks are processed on m_1.

V_2: The jobs in J whose both tasks are processed on m_2.

V_3: The jobs j_i in J whose task $(i,1)$ is processed on m_1 and task $(i,2)$ is processed on m_2.

We use the term *Johnson's Order* as a sequence of jobs that has the property that: *Job* J_i *precedes job* J_j *if* $\min\{p_{i1}, p_{j2}\} \leq \min\{p_{j1}, p_{i2}\}$ (see Johnson (1954) for motivation of this term). The following properties hold for the optimal schedule of the 2FFS environment.

Property 1 *There exists an optimal schedule where all jobs in* V_1 *follow the jobs in* V_3.

Property 2 *There exists an optimal schedule where all jobs in* V_2 *precede the jobs in* V_3.

Property 3 *There exists an optimal schedule where all jobs in V_3 are in Johnson's order.*

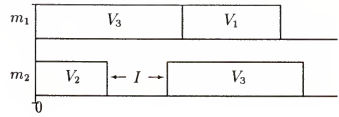
Property 4 *There exists an optimal schedule with no idle time on m_1.*

Properties 1 through 3 can be verified either by straightforward job interchange arguments or by simply observing that the 2FFS is a special case of the two machine job shop environment. In the job shop environment we have four classes of jobs: classes V_1, V_2, V_3, and a new class V_4 of jobs whose task $(i,1)$ is processed on machine m_2 and task $(i,2)$ is processed on m_1. Thus, the 2FFS is a two machine job shop with V_4 defined as an empty set. Property 4 is an obvious one motivated by the makespan minimization objective. We would like to note that when the jobs belonging to each of the classes V_1-V_4 have been prespecified, then the two machine job shop makespan minimization objective problem can be solved with the use of a variation of Johnson's algorithm in $\mathcal{O}(n \log n)$ time (see French (1982)). However, for scheduling the 2FFS the composition of jobs in each class has to be specified by the solution procedure. This results in an increase of the combinatorial complexity of the problem to that of an $\mathcal{NP}$-complete problem.

We observe that since there may be idle time on m_2 we can always shift jobs in V_2 to the left and shift jobs in V_3 to the right so that the resulting schedule processes all jobs in V_2 contiguously starting at time zero, followed by I periods of idle time which are thereafter followed by the jobs in V_3 also scheduled contiguously. Clearly, the above shifts can be done so that neither the makespan is increased nor feasibility is destroyed. Combining properties 1-4 with the previous observations, we can conclude that there exists an optimal solution for the 2FFS which has the structure depicted in Figure 4.3.

In the next section we will use the above properties to develop a pseudopolynomial dynamic programming algorithm for generating optimal schedules for the 2FFS.

Figure 4.3. The structure of an optimal solution for the 2FFS



4.3 A Dynamic Programming Algorithm for the 2FFS

4.3.1 Description of the DP Algorithm

Assume that the set $J = \{j_1, j_2, \ldots, j_n\}$ is ordered according to Johnson's order. Define the quantities:

$A_i = \sum_{j=1}^{i} p_i$ for $i = 1, 2, \ldots, n$ and

$f_i(t, x, id)$ = the makespan of an optimal schedule whose total processing time on m_1 of jobs in $V_3 \bigcap \{j_1, j_2, \ldots, j_i\}$ is t, the total processing time on m_1 of jobs in $V_1 \bigcap \{j_1, j_2, \ldots, j_i\}$ is x and the idle time on m_2 is id.

By definition, the values that the variables t, x, id can take on are:

$$t = 0, 1, \ldots, \sum_{i=1}^{n} p_{i1}, \quad x, \ id = 0, 1, \ldots, \sum_{i=1}^{n} p_i$$

Boundary Conditions: $f_0(0, 0, id) = id$ for $id = 0, 1, \ldots, \sum_{i=1}^{n} p_i$, and

$f_0(t, x, id) = \infty$ otherwise.

Recurrence Relation:

$$f_i(t, x, id) = \min \begin{cases} \max\{x + t, A_i - x - t + id, f_{i-1}(t, x - p_i, id)\} \\ \max\{x + t, A_i - x - t + id, f_{i-1}(t, x, p_i + id)\} \\ \max\{x + t, A_i - x - t + id, f_{i-1}(t - p_{i1}, x, id)\} \\ \quad if \quad t \leq A_i - x - t + id - p_{i2}. \end{cases} \tag{1}$$

Optimal Solution:

Let f_n^* denote the optimal solution to the dynamic program (1) with the above given boundary conditions. Then

$$f_n^* = \min \begin{cases} \min\{f_n(t, 0, id) : 0 \leq t \leq \sum_{i=1}^{n} p_{i1}, 0 \leq id \leq \sum_{i=1}^{n} p_i\} \\ \min\{f_n(t, x, id) : 0 \leq t \leq \sum_{i=1}^{n} p_{i1}, 1 \leq x \leq \sum_{i=1}^{n} p_i, 0 \leq id \leq \max_i p_i\}. \end{cases} \tag{2}$$

Justification:

At the i-th stage of the dynamic program, job j_i has to be scheduled. The first branch of the recursive relationship (1) corresponds to the case that j_i is scheduled to be processed in V_1, the second to the case that j_i is included in V_2 and the third to the case that j_i is scheduled to be processed in V_3. The first branch is meaningful only in case $f_{i-1}(t, x - p_i, id) < \infty$, the second one if $f_{i-1}(t, x, p_i + id) < \infty$ and the third if $f_{i-1}(t - p_{i1}, x, id) < \infty$. In addition, for the third branch we have to ensure feasibility of the schedule, i.e. the starting time of task $(i, 2)$ must be no earlier than the finishing time of task $(i, 1)$. Since the finishing time of $(i, 1)$ is t and the starting time of $(i, 2)$ is $A_i - x - t + id - p_{i2}$, the condition $t \leq A_i - x - t + id - p_{i2}$ guarantees the feasibility of the schedule.

At any stage of the dynamic program, the first machine finishes at time $x + t$ while the second one finishes at time $A_i - x - t + id$. The maximum of these two quantities determines the makespan of the partial schedule (i.e. the schedule of jobs $\{j_1, j_2, \ldots, j_i\}$). The minimum over the values obtained by the three branches of our recurrence relationship determines the value of $f_i(x, t, id)$, as in (2).

Relationship (2) implies that there exists an optimal schedule for which either the variable id takes on values from $1, 2, \ldots, \max_{1 \leq i \leq n} p_i$ or $x = 0$. This is because if $x > 0$ and $id > \max_{1 \leq i \leq n} p_i$, we can always schedule one of the jobs of V_1 to be processed during the idle time of m_2, thus reducing x to zero or making the idle time remaining on m_2 less than the quantity $\max_{1 \leq i \leq n} p_i$.

Complexity of the DP algorithm:

Define the quantity $MS = \sum_{i=1}^{n} p_i$. It is evident that none of the variables t, x, id can take on a value greater than MS. Also, the variable i can take on values $1, 2, \ldots, n$. Hence the number of states of the dynamic program is $\mathcal{O}(nMS^3)$. For each state it takes $\mathcal{O}(1)$ effort to evaluate relationship (1), and therefore the asymptotic complexity of our dynamic programming algorithm is $\mathcal{O}(nMS^3)$.

4.3.2 Computational Performance of the DP Algorithm

To test the performance of the dynamic program of the previous section we wrote a Turbo Pascal code and experimented on a 486 PC with math coprocessor. Within the constraint of 640 Kilobytes of base memory, we were able to solve problems up to 30 jobs. In Table 4.1 presented below, we report the average execution times for different processing time ratios. For example, a ratio $2:1$ means that on the average the processing time of a task $(i,1)$ is double the processing time of a task $(i,2)$. Also, the processing time of $(i,1)$ is chosen randomly from a uniform distribution on $[1,5]$ and the processing time of $(i,2)$ is chosen randomly from a uniform distribution on $[1,10]$.

We report execution times for the ratios $1:1$, $2:1,\ldots,8:1$ and their reciprocals. To accommodate for various processing time ratios (i.e. for high values of MS), we experimented with problem sizes of 10, 15 and 20. This results to 33 ratio/size combinations. For each combination, we report the average execution time (in seconds) over 5 randomly selected problems. Thus, we solved a total of 165 problems.

Table 4.1. Average execution times (in seconds) of the DP algorithm for 2FFS.

size/ratio	1 : 1	1 : 2	1 : 3	1 : 4	1 : 5	1 : 6	1 : 7	1 : 8
10	0.39	0.51	0.62	0.98	1.92	1.99	2.90	4.40
15	0.60	0.93	1.65	2.95	4.90	10.44	14.20	11.70
20	0.98	1.96	4.29	9.06	13.60	15.53	29.52	33.97
		2 : 1	3 : 1	4 : 1	5 : 1	6 : 1	7 : 1	8 : 1
10		0.89	1.15	1.42	2.52	2.15	3.19	4.01
15		1.28	2.13	3.53	4.29	7.03	10.84	16.66
20		2.60	4.26	8.71	15.04	17.38	30.44	41.25

In our experiments we observed that the execution time of each of the five problems generated for a specific ratio/size combination was very close to the corresponding average. This implies that for a given size/ratio combination the dynamic program

is robust with respect to the required computational effort.

It can be seen that the execution times increase as problem size increases and as the ratios increase. For ratios less than one (first three rows of the table) the execution times are smaller than those for the corresponding size execution times with ratios greater than one. When the problem size increases from 10 to 15, the execution time increases by a factor of 2 (for size=10 and ratio=1:2) to a factor of 4 (for size=10 and ratio=8:1). When the problem size increases from 15 to 20, the execution time increases by a factor of 2 (for size=15 and ratio=1:2) to a factor of 4 (for size=15 and ratio=7:1). However, all test problems were solved in less than 42 seconds. We conclude that the DP procedure can handle efficiently medium size problems in microcomputer supported environments.

4.4 Heuristic Algorithms for the 2FFS

4.4.1 A Modified Parallel Machine Heuristic

As we explained in Section 4.2.1, the flexible flowshop represents a generalization of the traditional flowshop and the identical parallel machine environment. Therefore, algorithms for the two machine flowshop problem (e.g. Johnson's Algorithm (JA), see Johnson (1954) or the two identical parallel machine environment (for a comprehensive review see Cheng and Sin (1990)) can be used as heuristics for our problem. In this section, we focus on modifying one of the parallel machine heuristics to better depict the characteristics of the 2FFS and subsequently perform better when applied to such environments. Our reasons for selecting this particular heuristic were its implementation simplicity, low complexity and very good average and worst case performance. We selected to modify the heuristic referred to as the RDM heuristic (see Cheng and Sin (1990)). This heuristic is described in pseudoalgorithmic format below:

Heuristic RDM:

Step 1. Order jobs in J at random.

Step 2. Pick the job at the top of the list J and load it to the first available machine.

Step 3. If J is empty stop, else goto Step 2.

In *RDM*, each job i is considered to consist of a single task that requires processing time p_i. If C_{RDM} is the makespan of the schedule on the two parallel machines obtained by application of *RDM* and C_{2P} is the optimal makespan value for the two parallel machine environment, then it is known that $C_{RDM}/C_{2P} \leq 3/2$ (see Cheng and Sin (1990)). Clearly the complexity of *RDM* is $\mathcal{O}(n)$. We will modify this heuristic so that the same properties (i.e. complexity $\mathcal{O}(n)$ and 1.5 worst case performance) hold true for the application of the heuristic to the 2FFS environment. We refer to this heuristic as the Modified RDM (MRDM) heuristic.

Heuristic MRDM:

Step 1. Apply *RDM* on the two parallel machine environment with processing times $p_i, i = 1, 2, \ldots, n$.

Step 2. Compute the latest job completion times C_1 and C_2, according to the resulting schedule from Step 1, of production stages m_1 and m_2 respectively. Let j_i be the last job scheduled on m_1 and j_j the last job scheduled on m_2.

Step 3. *If* $(C_1 \geq C_2)$ *and* $(C_1 - p_{i2} > C_2)$ *then*
schedule task $(i, 1)$ first on m_1, and
schedule task $(i, 2)$ last on m_2

Step 4. *If* $(C_1 < C_2)$ *and* $(p_{j1} < C_2 - C_1)$ *and* $(p_{j2} < C_2 - C_1)$ *then* schedule task $(j, 1)$ first on m_2, and

schedule task $(j,2)$ to start at time $\max\{C_1, p_{j1}\}$ on m_1.

All jobs previously scheduled on m_1 are rescheduled to be completely processed, on the same order as before, by m_2.

All jobs previously scheduled on m_2 are rescheduled to be completely processed, on the same order as before, by m_1.

Clearly *MRDM* is still $\mathcal{O}(n)$. Note that the solution obtained from *MRDM* is feasible to the 2FFS because for every job j_i the task $(i,1)$ by construction precedes task $(i,2)$. Actually, there is at most one job that has its processing split between the two production stages (see Step 4). We will show that if C_{MRDM} is the makespan of the 2FFS obtained by application of *MRDM* and C_{2FFS} is the optimal makespan value for the same environment, then $C_{MRDM}/C_{2FFS} \leq 3/2$.

The following additional notation will be needed for our further discussion. Let C^{2n}_{ALG} denote the makespan of the schedule obtained by algorithm ALG when each of the tasks $(i,1)$ and $(i,2)$ is considered to be a job by itself with processing time p_{i1} and p_{i2} respectively. ALG, can be either an optimal algorithm for the two parallel machine environment or the heuristic *RDM*. For example, C^{2n}_{RDM} denotes the makespan obtained by application of *RDM* to the two parallel machine environment with $2n$ jobs, i.e. when for each job i each of its two tasks is considered as a different job, and C^{2n}_{2P} denotes the makespan obtained by application of an optimal algorithm for the parallel machine environment.

The two propositions that follow are instrumental in providing a rigorous proof to the worst case bound of *MRDM*. Let S be a schedule obtained by application of *RDM* to the n job two parallel machine environment (i.e. the processing time of each job is $p_i, i = 1,2,\ldots,n$) and C_1, C_2 the completion times of the last jobs of production stages m_1, m_2 respectively in S. Assume that $C_1 \geq C_2$. Let j_i be the last job scheduled on m_1 by S. Then we can state the following.

Proposition 1 *If $C_1 - p_{i2} \leq C_2$ we can find a list L of the tasks $(i,1)$, $(i,2)$ $i = 1,2,\ldots,n$ so that RDM acting on L results to schedule S.*

Proof. The schedule S assigns a starting time to each job j_i in J. Therefore, S induces a starting time to each of the tasks $(i,1)$, $(i,2)$. In particular, if s is the starting time of j_i, then s is the starting time of $(i,1)$ and $s + p_{i1}$ is the starting time of $(i,2)$.

Let L be the list obtained by the nondecreasing order of the starting times of the $2n$ tasks. Since $C_1 - p_{i2} \leq C_2$, we can apply the minimum loaded machine rule with respect to L to choose the production stage to allocate the next task on the list so that the resulting schedule is S. To do this, whenever both stages are equally loaded, we have to schedule the task at the top of L to the stage that S scheduled that task. Therefore, when *RDM* acts on L with appropriate tie-breaking, the resulting schedule is S.□

Clearly, the Proposition holds for the symmetric case where $C_2 > C_1, C_2 - p_{i2} \leq C_1$ and j_i is the last job scheduled on m_2 in S.

Proposition 2 *There exists a list L for which either $C_{MRDM} = C_{2FFS}$ or $C_{MRDM} \leq C^{2n}_{RDM}$.*

Proof. Let S be the sequence obtained at the end of Step 1 of *MRDM* and C_1, C_2 the completion times of the last jobs of m_1, m_2 respectively in S. Also, let j_i be the last job scheduled on m_1 by S and j_j the last job scheduled on m_2 by S.

Case 1: $C_1 \geq C_2$

subcase i: $C_1 - p_{i2} \leq C_2$. According to Proposition 1, there exists a list L such that $C^{2n}_{RDM} = C_1$. In this subcase $C_{MRDM} = C_1$ and therefore for this list L it holds that $C_{MRDM} = C^{2n}_{RDM}$.

subcase ii: $C_1 - p_{i2} > C_2$. Then, $C_1 - p_i \leq C_2$ because jobs in S are scheduled according to the minimum loaded machine rule to allocate tasks to production stages. Now we can apply Proposition 1 on the $2n-1$ tasks excluding $(i,2)$. Let L' be the

corresponding list. When *RDM* is applied to the ordered list $L = (L', (i,2))$, the resulting schedule is identical to S with the only difference that $(i,2)$ is scheduled last on m_2 instead of last on m_1. Thus, for the sequence L, $C^{2n}_{RDM} = \max\{C_1 - p_{i2}, C_2 + p_{i2}\}$. In this subcase, *MRDM* schedules $(i,1)$ first on m_1 and $(i,2)$ last on m_2. If $p_{i1} \leq C_2$ then no idle time needs to be inserted in m_2 and therefore $C_{MRDM} = \max\{C_1 - p_{i2}, C_2 + p_{i2}\}$. Thus, $C_{MRDM} = C^{2n}_{RDM}$. If $p_{i1} > C_2$ then $p_{i1} - C_2$ units of idle time need to be inserted in m_2 and therefore $C_{MRDM} = \max\{C_1 - p_{i2}, p_i\}$. If $C_{MRDM} = p_i$ then *MRDM* produces an optimal schedule for the 2FFS (since $C_{2FFS} \geq p_i$ for $i = 1, 2, \ldots, n$); otherwise, $C_{MRDM} \leq C^{2n}_{RDM}$ (because $C_1 - p_{i2} \leq \max\{C_1 - p_{i2}, C_2 + p_{i2}\}$).

Case 2: $C_1 < C_2$

subcase i: $p_{j1} \geq C_2 - C_1$ or $p_{j2} \geq C_2 - C_1$. If $p_{j2} \geq C_2 - C_1$ then by Proposition 1 there exists a list L of the $2n$ tasks such that *RDM* acting on L results to S and therefore $C_{MRDM} = C^{2n}_{RDM}$. Similarly, if $p_{j1} \geq C_2 - C_1$, but $p_{j2} < C_2 - C_1$, then we can interchange the positions of $(i,2)$ and $(i,1)$ in S and by the previous argument conclude that $C_{MRDM} = C^{2n}_{RDM}$.

subcase ii: $p_{j1} < C_2 - C_1$ and $p_{j2} < C_2 - C_1$. Then $C_2 - p_j \leq C_1$ because S is obtained by using the first available machine rule to allocate tasks to production stages. Then, *MRDM* schedules the task $(j,1)$ first on m_2 and task $(j,2)$ last on m_1. Then it swaps the loads of m_1, m_2. Therefore this subcase is identical to Case 1, subcase ii. Hence, $C_{MRDM} = C_{2FFS}$ or $C_{MRDM} \leq C^{2n}_{RDM}$. □

Lemma 3 $C^{2n}_{2P} \leq C_{2FFS} \leq C_{MRDM}$.

Proof. Every solution that is feasible for the 2FFS is also feasible for the $2n$ job two parallel machine environment, but not the other way around. Therefore, $C^{2n}_{2P} \leq C_{2FFS}$. Also, the solution obtained by *MRDM* is by construction feasible to the 2FFS. Therefore $C_{2FFS} \leq C_{MRDM}$. □

Theorem 8 $\frac{C_{MRDM}}{C_{2FFS}} \leq \frac{3}{2}$ *and the bound is tight.*

Proof. If $C_{MRDM} = C_{2FFS}$ then the theorem holds trivially. Else, we know from Proposition 2 that there exists list L such that $C_{MRDM} \leq C^{2n}_{RDM}$. If Lemma 3 is applied for L we obtain that

$$C^{2n}_{2P} \leq C_{2FFS} \leq C^{2n}_{RDM}$$

and therefore

$$\frac{C^{2n}_{RDM}}{C_{2FFS}} \leq \frac{C^{2n}_{RDM}}{C^{2n}_{2P}} \leq \frac{3}{2}$$

and since for this L we have $C_{MRDM} \leq C^{2n}_{RDM}$, then

$$\frac{C_{MRDM}}{C_{2FFS}} \leq \frac{3}{2}.$$

This bound is tight for the two parallel machine environment (see Cheng and Sin (1990)). Since the two parallel machine problem is a special case of the 2FFS, this bound is tight for our problem as well. □

As we will report below, the average performance of *MRDM* is even more pronounced since it produces near optimal solutions to a large number of randomly generated problems. Table 4.2 reports the relative percentage error $\frac{C_{MRDM}-C_{2FFS}}{C_{2FFS}}100\%$ for various problem sizes (i.e. number of jobs processed) and processing time ratios (for definition of a processing time ratio see Section 4.3.2). The performance of the heuristic is tested for the processing time ratios 1:1, 1:2, 1:3, 1:4 and their reciprocals. Problems of size 10, 15 and 20 were generated for each processing time ratio, thus resulting to a total 27 size/ratio combinations. For each combination, we report the average performance of *MRDM* over 5 randomly generated problems. Thus a total of 135 problems were solved.

Table 4.2 suggests that MRDM performs extremely well. The overall average performance is only 1.02% away from the optimal. From the table we conclude that

Table 4.2. Deviation of *MRDM* from optimality.

$\frac{C_{MRDM}-C_{2FFS}}{C_{2FFS}}100\%$					
size/ratio	1 : 1	1 : 2	1 : 3	1 : 4	Average
10	1.5	2.5	5.2	2.1	2.82
15	1.1	1.4	0.78	0.42	0.92
20	0.79	0.71	1.2	0.00	0.67
Average	1.13	1.53	2.39	0.84	
		2 : 1	3 : 1	4 : 1	
10		1.5	0.5	1.3	1.1
15		0.36	0.82	0.35	0.51
20		0.34	0.00	0.00	0.11
Average		0.73	0.44	0.55	Overall: 1.02

the relative error reduces as the problem size increases. Also, the heuristic performs better for processing time ratios greater than one. We observed that MRDM solved a great number of problems optimally, especially when the problem size increased. Since the average error of MRDM is small, we feel confident in recommending the heuristic as an adequate tool for scheduling the 2FFS environment with a large number of jobs.

4.4.2 Performance of Johnson's Algorithm as a Heuristic for the 2FFS

As previously mentioned, Johnson's algorithm (JA) can be thought of as a heuristic procedure for the 2FFS. Below we perform a worst case analysis of the algorithm when applied to the 2FFS environment.

Let us refer to the 2FFS depicted in Figure 4.1. A different way to interpret the operation of that environment is the following. The processing requirement of p_{i1} units will be either performed on M_a or on M_b, and the processing requirement of the subsequent p_{i2} units will be either performed on M_c or on M_d. We introduce the following auxiliary problem. Replace the machines M_a, M_b by a dummy machine M_1 and replace the processing time requirement of p_{i1} units on one of M_a, M_b by the requirement of $1/2p_{i1}$ units on M_1. Also, replace the machines M_c, M_d by a

dummy machine M_2 and replace the processing time requirement of p_{i2} units on one of M_c, M_d by the requirement of $1/2p_{i2}$ units on M_2. This way we define an auxiliary 2-machine flowshop problem on M_1, M_2.

Lemma 4 *Let C_{JA} be the completion time found by JA for the auxiliary problem. Then, $C_{JA} \leq C_{2FFS}$.*

Proof. Let S be an optimal sequence for *2FFS* (the processing times for this problem are (p_{i1}, p_{i2})). Let $t_1, t_2, \ldots, t_n$ be the nondecreasing order of completion times of the tasks $(i, 1)$ on M_a, M_b.

We will show that the optimal solution to the auxiliary problem is a lower bound for C_{2FFS}, i.e. $C_{JA} \leq C_{2FFS}$. We do this by showing that the sequence S which is not necessarily optimal for the auxiliary problem, achieves a makespan $C_S \leq C_{2FFS}$, where C_S is the makespan of sequence S applied to the auxiliary problem.

Indeed, for every $i = 1, 2, \ldots, n$, the finishing time of j_i (i.e. the i-th job in S) on M_1, say C'_i, is $C'_i = 1/2(t_i + t_{i-1})$. Since $t_{i-1} \leq t_i$ implies $C'_i \leq t_i$, we can infer that the finishing time of task $(i, 1)$ on M_1 is less than or equal to the finishing time of the same task on M_a or M_b. A similar argument shows that the completion time of task $(i, 2)$ on M_2 is less than or equal to the completion time of the same task on M_c or M_d. This last fact shows that the total idle time inserted on M_2 for the auxiliary problem is no more than the idle time inserted on any of M_c or M_d for the *2FFS*. Thus, $C_S \leq C_{2FFS}$.

Since S is not necessarily optimal for the auxiliary problem, we have that $C_{JA} \leq C_S \leq C_{2FFS}$. This completes the proof of the lemma. □

Let C_{FS} be the makespan obtained by Johnson's algorithm when applied on the traditional flowshop (FS) (i.e. task $(i, 1)$ of job i is performed on stage 1 and task $(i, 2)$ on stage 2). Then we can state the following.

Theorem 9 $\frac{C_{FS}}{C_{2FFS}} \leq 2$ *and the bound is tight.*

Proof. Clearly, $C_{FS} \geq C_{2FFS}$ because the *FS* is a subsystem of the *2FFS*. In Lemma 4 we showed that the makespan C of the auxiliary problem is $C \leq C_{2FFS}$. Also, the sum of all processing times of the auxiliary problem cannot exceed C, i.e. $\frac{1}{2}(\sum_i p_{i1} + \sum_i p_{i2}) \leq C$.

If we obtain a solution to the auxiliary problem, we can construct a solution for *2FFS* by replacing the processing time of $\frac{1}{2}p_{i1}$ by p_{i1} and scheduling the machine M_a of *2FFS* to process the load of machine M_1 in *FS* and machine M_d to process the load of M_2. In the resulting solution for the *2FFS* the machines M_b, M_c will not be utilized and the makespan will be $C + \frac{1}{2}(p_{i1} + p_{i2}) \leq 2C$. Thus, $C_{2FFS} \leq 2C$.

In addition $C \leq C_{FS}$ because the processing times of the jobs for the auxiliary problem are less than or equal to the processing times of the jobs for *FS*. Hence, $C \leq C_{FS}$ and therefore $\frac{C_{2FFS}}{C_{FS}} \leq 2$.

This error bound is tight. To see this consider an instance with $n+1$ jobs with processing times $p_{i1} = 1$ for $i = 1, 2, \ldots, n$, $p_{n+1,1} = n$ and $p_{i2} = 0$ for $i = 1, 2, \ldots, n+1$. One can easily see that $C_{2FFS} = n$ while $C_{FS} = 2n$. Thus for this example $\frac{C_{FS}}{C_{2FFS}} = 2$. □

In effect, this Theorem shows that Johnson's algorithm provides a heuristic for the *2FFS* with an error bound of 2. As we will report later (see Section 4.5, Table 4.3), the average makespan performance of JA is 13% away from the optimal makespan of the 2FFS.

4.5 Flexible Flowshops versus Traditional Flowshops and Parallel Machines

In this section we study the effects of important characteristics of the production environment (in particular, stage workloads and variation among processing times of the jobs at each stage) on three different configurations (we are restricting ourselves to the two-stage cases): flowshop (as traditionally defined, with all jobs following the same routing of stages and a nonempty prespecified subset of jobs requiring processing

at both stages of the system), identical parallel production stages (machines), and flexible flowshop (as defined in Section 4.2.1). For all environments job preemption is not allowed.

More precise definitions of the characteristics of the production environment we considered, are the following:

(a) Stage Workload Ratio (LR): it reflects the ratio of workloads at the two stages of the traditional flowshop environment, i.e.

$$LR = \frac{\sum_{i=1}^{n} p_{i1}}{\sum_{i=1}^{n} p_{i2}}$$

(b) Coefficient of Variation of processing times at stage k, CV_k $(k = 1, 2)$: it reflects the variability of processing times of jobs at a specific stage k $(k = 1, 2)$ (as a reminder, the processing time of each job is deterministic, and the variability we are referring to is among processing times of jobs at a stage), i.e.

$$CV_k = \frac{\sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(p_{ik} - \bar{p_k})^2}}{\bar{p_k}}, \; where \; \bar{p_k} = \frac{\sum_{i=1}^{n} p_{ik}}{n}, \; k = 1, 2.$$

In Table 4.3 we report the results of our computational experiments. We generated random scheduling problems for a flowshop environment with 15 jobs within prespecified ranges for the workload ratio (for balanced workloads the ratio LR $\in [0.95, 1.05]$, for unbalanced workloads -lower workload on stage 1- $LR \in [0.6, 0.8]$, and for unbalanced workloads -lower workload on stage 2- $LR \in [1.2, 1.4]$) and the coefficients of variation (for lower processing time variability at stage k, $CV_k \in [0.5, 0.7]$, for medium processing time variability at stage k, $CV_k \in [0.9, 1.1]$, and for high processing time variability at stage k, $CV_k \in [1.2, 1.4]$). There is a total of 27 combinations of ranges for the various parameters, and for each combination 5 random scheduling problems were generated. Thus a total of 135 test problems were solved. For each combination of parameters, we report the average over the 5 test problems within that range of the following two quantities:

i) Percentage deviation of the makespan performance of the flowshop from the optimal makespan performance of the flexible flowshop, i.e. $\frac{C_{FS}-C_{2FFS}}{C_{2FFS}}100\%$, and

ii) Percentage deviation of the makespan performance of the schedule resulting from the application of an optimal algorithm for the parallel machine environment (see Cheng and Sin (1990)) from the optimal makespan performance of the flexible flowshop, i.e. $\frac{C_{2P}-C_{2FFS}}{C_{2FFS}}100\%$.

Below, we summarize and interpret the results of Table 4.3. We start by comparing the performance of 2FFS environments with traditional flowshops.

Table 4.3. Performance Comparison of Two Stage Flexible Flowshops with traditional Flowshops and Parallel Machine Environments

Workload Ratio LR	Coefficient of Variation CV_1	Coefficient of Variation CV_2					
		[0.5, 0.7]		[0.9, 1.1]		[1.5, 2]	
[0.6, 0.8]	[0.5, 0.7]	26	2.4	22	4.6	16.2	2.3
	[0.9, 1.1]	20	2.8	14	1.9	17.6	1.7
	[1.5, 2]	19.4	1.5	20	1.1	15.3	0.9
[0.9, 1.1]	[0.5, 0.7]	6.7	2.6	5.8	2.3	4	2.1
	[0.9, 1.1]	4.5	1.7	4.1	0.6	2	1.3
	[1.5, 2]	4	0.8	4	0.4	1.8	0.6
[1.2, 1.4]	[0.5, 0.7]	18	2	17	1.8	16.3	1.2
	[0.9, 1.1]	22	7.9	23	8.6	14.6	5.1
	[1.5, 2]	11.7	1.1	16	0.8	13.7	0.6

1a) The flexible flowshop environment significantly outperforms traditional flowshops. The makespan performance improvements are in the range of 2-26%. The differences are more pronounced for unbalanced workload flowshop environments. Specifically, we have that the average percentage deviation in performance of the two environments is 4.1% for balanced workloads across the two stages, 18.9% for unbalanced workloads with lower workload on stage 1 and 16.9% for unbalanced workloads with lower workload on stage 2.

1b) For a balanced workload flowshop environment with deterministic processing times, and in the absence of a bottleneck process, the scheduling problem becomes of less importance. The processing time variations for balanced workload environments affect somewhat the performance of the flowshop, and actually the flowshop performance improves as the CV_1 and CV_2 increase. For flowshop environments with large variance in the processing times of the jobs (i.e. many small and few large jobs) the logic of the Johnson's order efficiently schedules the jobs by pushing small jobs early in the first stage and late in the second stage, without leaving significant idle times on the stages. This can be easily seen at the Gantt charts of the optimal flowshop and 2FFS environments of a representative example with balanced workloads (see Figure 4.4).

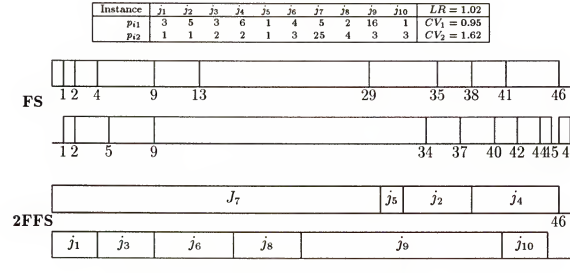


Figure 4.4. An example with balanced workloads

1c) For unbalanced workload environments, particularly when the second stage is the overloaded one, and thus acts as the bottleneck, scheduling of the first stage becomes crucial in the flowshop environment. In this case processing time variation, in particular in the first stage, helps the Johnson order to find and schedule an adequate number of small jobs in the first stage, and thus reduce any forced idle

Instance	j_1	j_2	j_3	j_4	j_5	j_6	j_7	j_8	j_9	j_{10}	$LR = 0.7$
p_{i1}	1	2	1	1	2	2	2	1	15	1	$CV_1 = 1.54$
p_{i2}	2	3	2	9	3	1	12	4	1	3	$CV_2 = 0.9$

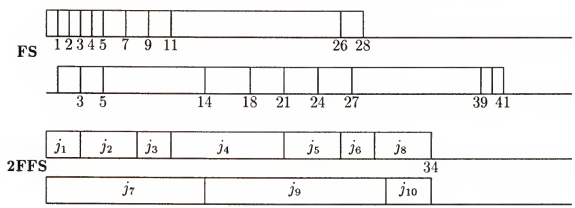


Figure 4.5. An example with unbalanced workloads

Instance	j_1	j_2	j_3	j_4	j_5	j_6	j_7	j_8	j_9	j_{10}	$LR = 1.34$
p_{i1}	1	8	2	2	16	3	0	12	1	10	$CV_1 = 1.02$
p_{i2}	2	1	3	0	4	2	22	1	3	3	$CV_2 = 1.56$

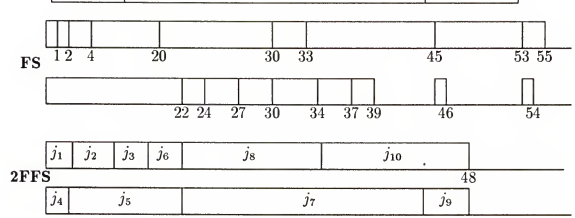


Figure 4.6. An example with significant processing time variability

time on the second stage because of stage precedence constraints. This is depicted in the Gantt charts for the flowshop and the 2FFS environments of a representative example of this case (see Figure 4.5). When the first stage is the overloaded one, and particularly when there exists significant processing time variation in the second stage, the stage precedence constraints are rather weak. The best performance of a flowshop in such cases occurs when the second stage has rather insignificant idle times as it is continuously fed from the first stage. This is highly likely to happen in the presence of significant processing time variability in the second stage. We depict such an example in Figure 4.6.

We now proceed to compare the performance of the 2FFS with two parallel machine environments:

2a) The 2FFS environment performs better than the 2P environment. The makespan performance improvements are in the range of 0.6-8.6%. In general, the parallel machine environment tends to perform quite close to the performance of the 2FFS. The differences in performance are more pronounced for unbalanced workload environments, since for those environments the 2FFS better exploits the across stages processing flexibility (this is consistent with our previous discussion on the comparison of the FS and 2FFS environments). Specifically we have that the average percentage deviation in performance of the two environments is 1.4% for balanced workloads across the two stages, 2.13% for balanced workloads with lower workload on stage 1 and 3.23% for unbalanced workloads with lower workload on stage 2. (As a reminder, the workload definition relates to the two stages of a traditional flowshop).

2b) It is rather difficult to come up with any conclusive arguments on the effects of the processing time variations on the performance of parallel machine environments. Coming up with simple explanations for the parallel machine environment, as we were able to do for the flowshop environment, is in part prohibited by the absence of simple polynomial rules in generating optimal schedules for the parallel machine

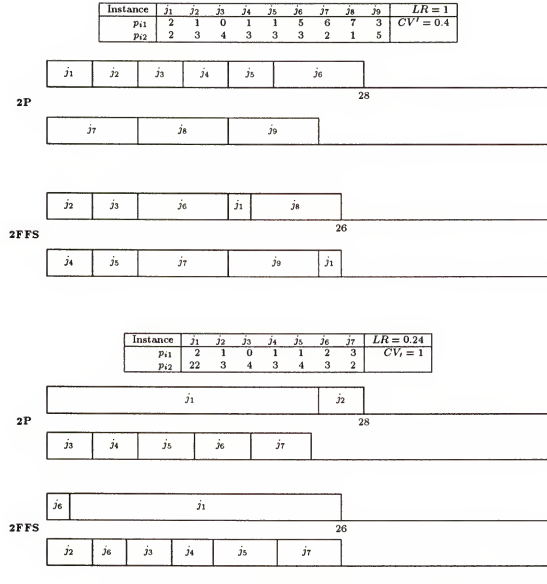


Figure 4.7. Examples where 2FFS significantly outperforms 2P

environment. In general, the 2FFS tends to use the processing time flexibility across stages advantageously to achieve makespan values that are close to $\frac{1}{2}\sum_i p_i$, which is a lower bound makespan performance for all of the above described environments. For all cases that the parallel machine environments performed poorly as compared to the 2FFS, the difference $C_{2P} - \frac{1}{2}\sum_i p_i$ was significant. Such differences are usually driven by the processing times of all jobs and cannot be adequately explained by the use of the three parameters LR, CV_1 and CV_2 (or actually any other obvious choice of a small set of aggregate parameters of the problem we attempted to come up). To demonstrate this point we provide two examples (see Gantt charts of the 2P and 2FFS environments of the examples in Figure 4.7) of relatively poor performance of the parallel machine environment with almost diametrically different values of workload and processing time variation parameters. We use, as more appropriate for these examples, the processing time coefficient of variation CV' defined as

$$CV' = \frac{\sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(p_i - \bar{p})^2}}{\bar{p}}, \; where \; \bar{p} = \frac{\sum_{i=1}^{n} p_i}{n}.$$

In the first example there is unbalanced workload and significant processing time variation, as expressed by CV' (non-favorable situation for the 2P), and in the second example there is balanced workload (non-favorable situation for the 2FFS) and small processing time variation (a rather favorable situation for 2P). In both examples the specific values of the job processing times lead to a parallel machine makespan significantly away from the lower bound $\frac{1}{2}\sum_i p_i$, while the 2FFS is able to achieve the above lower bound.

4.6 Conclusion

Making process choice decisions in environments with significant processing flexibility (mostly due to the presence of flexible machines) is a difficult task. In this

chapter we compared the throughput performance of three different process configurations for production systems with two flexible production stages (a production stage might be a machine or a manufacturing cell). The traditional way to operate these systems is one of the following:

(a) As two parallel machine environments, with the two multipurpose/multitool flexible machines being identically tooled. Then the production planning problem is that of allocating the jobs to the machines.

(b) As a flowshop environment, with the two flexible machines being tooled in a non-identical fashion. Part of the production task of each job is performed on each of the machines. Thus, the flexible machines become the production stages of a traditional flowshop with a unidirectional routing from the 'upstream' machine to the 'downstream' one. For this environment the production planning challenge becomes that of appropriately scheduling the two machines.

In this chapter we suggest a different way to operate such flexible systems, i.e.

(c) A flexible flowshop environment, with the two flexible machines being tooled in a non-identical fashion. The processing flexibility across stages is exploited by allowing some of the jobs to be processed completely on only one of the machines while for others the production task is split between the two machines. The production planning task for the flexible flowshop is somewhat more involved as the production tasks of each job have to be allocated to the two machines and then be appropriately scheduled. The additional complexity is compensated by the benefit of improved throughput performance, in particular as compared with the performance of traditional flowshops. Parallel machine implementations tend to have a uniformly good performance over a variety of production environments, though lower than that of the 2FFS, and also enjoy the advantage of a simplified routing (and thus material handling cost) control structure. To keep our summary of the results of the chapter at a brief level, we refer the reader to a table of comparative results for the three

environments (see Table 4.4). These results, mostly supported by the computational study of Section 4.5, can be used by the production manager as guidelines for making process choice decisions in flexible manufacturing environments.

Table 4.4. Comparative results for alternative two stage process configurations: parallel machines (2P), flowshop (FS), and flexible flowshop (2FFS)

Dimensions of Comparison	Process Configuration		
	Parallel Machines (2P)	Flowshop (FS)	Flexible Flowshop (2FFS)
Production Planning Task	Allocation of jobs to machines	Scheduling of two machines	Allocation of tasks to machines and scheduling of machines
Complexity of Optimal Algorithm	$\mathcal{O}(nMS)$	$\mathcal{O}(n \log n)$	$\mathcal{O}(nMS^3)$
Routing Control	Not needed	Unidirectional Routing	Unidirectional Routing
Throughput Performance	2.25% lower than 2FFS	13% lower than 2FFS	Highest
Characteristics of the Production Environment that favors its Throughput Performance	Uniform performance across a variety of environmental characteristics	Balanced workloads and large processing time variations at each machine	Unbalanced workloads and significant processing time variations in each machine

CHAPTER 5

THE TWO-STAGE FLEXIBLE JOBSHOP SCHEDULING PROBLEM

5.1 Introduction

This chapter addresses a scheduling problem for a two stage production environment with significant processing flexibility across production stages. We refer to this environment as a two stage 'flexible' job shop; for convenience in our further discussion we use the abbreviated term 2FJS. A job can be either completely processed at one of the stages or its processing is split in a prespecified fashion between the two stages. For the latter case the routing can be in any of two feasible flow directions between the two stages. If we restrict the routing from an 'upstream' stage to a 'downstream' stage, then the resulting environment is referred to as the 'flexible' flowshop environment. We assume the presence of adequate storage space between the two stages.

The flexible flowshop environment just described is frequently encountered in flexible manufacturing systems (FMS) (see Maleki (1991) for detailed description of FMS's), where each production stage might be either a flexible machine or a flexible manufacturing cell. Flexible machines are designed with large tool magazines that can accommodate a great variety of tools (up to 120 different tools). Depending on the tool assignment, the flexible machine can perform a number of different tasks and in many cases can complete all the required operations of a job. The processing flexibility of such machines is accurately depicted in our flexible job shop description as the operations of a job could be performed by either of the two machines or the

operations of the job could be split between the two machines. Flexible manufacturing cells (see Luggen (1991)) consist of a set of flexible machines, with not necessarily all of them being identically tooled, able to perform a variety of operations. For an environment with two flexible manufacturing cells, our flexible job shop problem amounts to deciding on how to allocate a set of jobs between the two cells, with the additional option of some jobs allowed to be split between the two cells. If both of the feasible routings between the two cells are allowed, then the environment is operated as a job shop. Operating the environment as a flowshop requires the use of a fixed routing from an 'upstream' cell to a 'downstream' cell for cases that the job is split between the two cells.

The problem can be also motivated for environments with dedicated parallel lines allowed to share resources for the processing of some jobs. Such an environment is described in Figure 5.1.

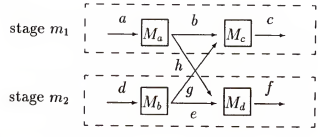


Figure 5.1. A Flexible Job shop Implementation in a Parallel Line Environment

Each one of the production stages is a multistation line. A job can be either processed completely on one of the lines or the first few processing steps are performed on line m_1 (line m_2) and the remaining steps are processed on line m_2 (line m_1). Such flexible job shop environments can be found in electronic assembly line environments, where the flexibility of the automatic component placement machines allows the manager to operate the lines in a dedicated or shared resources mode.

The study of scheduling problems in flexible manufacturing and assembly systems has attracted significant attention in recent years (see Shanker and Tzen (1985),

Shriskandarajah and Ladet (1986), Shriskandarajah and Sethi (1989), Stecke (1992), Wittrock (1988)) due to the importance of such systems for small-to-medium batch manufacturing. To the best of our knowledge, the 'flexible' job shop environment, as described in this chapter, has not been studied. Part of the above research literature evaluates the performance of algorithms which schedule parts through a manufacturing system referred to also as a 'flexible flowshop'. However, the use of the term in this chapter is different than that in the above literature. The previously described in the literature 'flexible flowshop' environments are multistage production environments with multiple identical parallel machines at each stage. The flexibility of these environments is derived from the ability to process a job on anyone of the parallel machines at a stage, i.e. it is processing flexibility within a production stage. The flexibility of our described environments ('flexible' flowshop and 'flexible' job shop) is a consequence of the fact that the job, or a part of it, could be processed on anyone of the production stages i.e. it is processing flexibility across production stages.

In this chapter, we concentrate on the scheduling problem of two stage flexible job shops with makespan minimization objective. A formal definition of the problem and properties of optimal 2FJS schedules are provided in Section 5.2. We present a pseudopolynomial dynamic programming algorithm for the solution of the problem in Section 5.3.

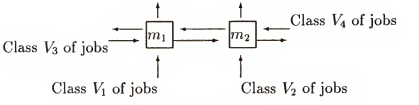
5.2 The Two Stage Flexible Job Shop Problem

5.2.1 Problem Description

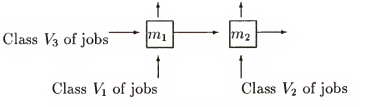
A set of jobs $J = \{j_1, j_2, \ldots, j_n\}$ is given and every job j_i consists of two tasks, namely $(i,1)$ and $(i,2)$, with processing time requirements p_{i1} and p_{i2} respectively. All jobs are assumed to be available at time zero and no preemption is allowed for the tasks. The jobs in J are to be processed in a two stage job shop which has the following type of processing flexibility. Every job j_i can be processed entirely on the

upstream stage m_1 for $p_i = p_{i1} + p_{i2}$ periods or entirely in the downstream stage m_2 for p_i periods or the two distinct tasks of the job are allocated to two different machines. For the latter case there are two possibilities: task $(i,1)$ is processed on m_1 for p_{i1} periods and task $(i,2)$ is processed on m_2 for p_{i2} periods or task $(i,1)$ is processed on m_2 for p_{i1} periods and task $(i,2)$ is processed on m_1 for p_{i2} periods. If we are going to restrict the operation of our environment to that of a flowshop, then for the latter case of job splitting between the two machines we allow only processing of task $(i,1)$ on machine m_1 and task $(i,2)$ on machine m_2 (without loss of generality, machine m_1 is the 'upstream' stage and machine m_2 is the 'downstream' stage). For maximizing throughput as well as machine utilization of the system, we are interested in sequencing the jobs in J so that the resulting schedule on the two stage flexible job shop minimizes the makespan.

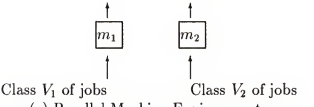
It is of interest to note that the above described flexible job shop represents a generalization of the traditional flowshop, job shop and identical parallel machine environments. The flexible job shop is an environment with four job classes; class V_1 of jobs require processing only on m_1, class V_2 of jobs require processing only on m_2, class V_3 of jobs require processing first at m_1 and then visit for further processing m_2, and class V_3 of jobs first get processed at m_2 and then at m_1 (see Figure (5.2a)). The four classes of jobs, can be also recognized at the flexible job shop environment described in Figure 5.1: class V_1 of jobs follow the route $a \rightarrow b \rightarrow c$, class V_2 of jobs follow the route $d \rightarrow e \rightarrow f$, class V_3 of jobs follow the route $a \rightarrow h \rightarrow f$, and class V_4 of jobs follow the route $d \rightarrow g \rightarrow c$. The flexible flowshop environment is an environment with no jobs in class V_4 (see Figure (5.2b)), the parallel machine environment is an environment with only classes V_1 and V_2 of jobs (see Figure (5.2c)), and the traditional flowshop can be thought of as an environment with only class V_3 of jobs (see Figure (5.2c)). The traditional job shop scheduling literature addresses the scheduling of the jobs given the composition of the classes V_1, V_2, V_3 and V_4.



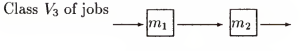
(a) Flexible and Traditional Job shop



(b) Flexible Flowshop



(c) Parallel Machine Environment



(d) Traditional Flowshop

Figure 5.2. The Flexible Job shop as a Generalization of the Flexible Flowshop, Parallel Identical Machines, Traditional Flowshop and Job shop Environments

Observe that in the case that $p_{i2} = 0$ for all $j_i \in J$, our flexible job shop problem reduces to the problem of minimizing makespan on two identical parallel machines. This parallel machine scheduling problem is known to be ordinary $\mathcal{NP}$-complete (see Garey and Johnson (1979)), and therefore our problem, since it contains the above problem as a special case, is $\mathcal{NP}$-complete as well.

5.2.2 Properties of Optimal 2FJS Schedules

In this subsection we identify a number of optimal properties for 2FJS schedules. First, given a solution S for 2FJS we partition the set of jobs J into the same four classes as in Figure (5.2a):

V_1: The jobs in J whose both tasks are processed on m_1.

V_2: The jobs in J whose both tasks are processed on m_2.

V_3: The jobs in J whose task $(i,1)$ is processed on m_1 and task $(i,2)$ is processed on m_2.

V_4: The jobs in J whose task $(i,1)$ is processed on m_2 and task $(i,2)$ is processed on m_1.

We use the term *Johnson's Order* as a sequence of jobs that has the property that: *Job* J_i *precedes job* J_j *if* $\min\{p_{i1}, p_{j2}\} \leq \min\{p_{j1}, p_{i2}\}$ (see Johnson (1954) for motivation of this term). The following properties hold for the optimal schedule of the 2FJS environment. These properties follow by simple interchange arguments that can easily inferred from optimal properties of traditional two stage flowshop and job shop environments as documented in scheduling textbooks (French (1982)).

Property 5 *There exists an optimal schedule where jobs in* V_3 *are scheduled before any other class of jobs on* m_1 *and after any other class of jobs on* m_2.

Property 6 *There exists an optimal schedule where jobs in V_4 are scheduled before any other class of jobs on m_2 and after any other class of jobs on m_1.*

Property 7 *There exists an optimal schedule where all jobs in V_3 are in Johnson's order.*

Property 8 *There exists an optimal schedule where all jobs in V_4 are in Johnson's order.*

The following lemma describes an interesting additional property (Property 5) of 2FJS schedules.

Lemma 5 *There exists an optimal schedule for which m_1 is the only machine that might have positive idle time and that idle time does not exceed the quantity $p_{\max} = \max_i\{p_{i1} + p_{i2}\}$.*

Proof. Observe first that we can always construct an optimal schedule such that if there is idle time on m_1 it is between job classes V_1 and V_4 and if there is idle time on m_2 it is between job classes V_2 and V_3. Using properties 1-4 and the above observation, the structure of an optimal schedule is depicted in Figure 5.3. It is quite easy to observe that $id_1 > 0$ only if $C_4^2 > C_{31}^1$ and $id_1 > 0$ only if $C_3^1 > C_{42}^2$. However, the above two conditions cannot hold at the same time since $C_4^2 > C_{31}^1$ implies $C_{42}^2 > C_4^2 > C_{31}^1 > C_{32}^1$ and similarly $C_3^1 > C_{42}^2$ implies $C_4^2 < C_{42}^2 < C_3^1 > C_{31}^1$. If $C_3^1 > C_{42}^2$ and thus there is idle time on machine m_2, then since m_1 and m_2 are identical in terms of their processing speeds we can interchange the role of m_1 and m_2 and obtain a schedule that has idle time on m_1. Thus, we can always assume that there exists an optimal schedule with the structure depicted in Figure 5.4. Observe now that if $id > p_{\max}$ and $V_2 \neq \emptyset$ we can always replace idle time on m_1 with at least one of the tasks p_{i1} assigned for processing on m_2. The case $V_4 = V_2 = \emptyset$ can always be transformed to an equivalent schedule in terms of makespan with $V_2 = \emptyset$

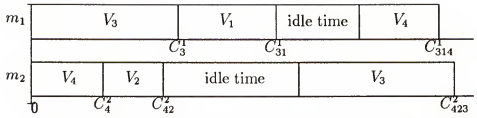


Figure 5.3. Supporting figure in the proof of Lemma 5

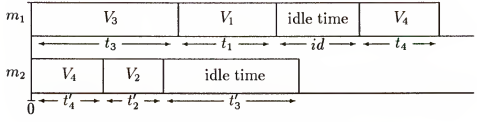


Figure 5.4. Structure of an optimal 2FJS schedule

(by allowing some of the jobs in V_3 to be completely processed on m_2). □

In the next section we will use the above properties to develop a pseudopolynomial dynamic programming algorithm for generating optimal schedules for the 2FJS.

5.3 A Dynamic Programming Algorithm for the 2FFS

5.3.1 Description of the DP Algorithm

Assume that the set $J = \{j_1, j_2, \ldots, j_n\}$ is ordered according to Johnson's order. Define the quantities:

$p_i = p_{i1} + p_{i2}$ for $i = 1, 2, \ldots, n$,

$MS = \sum_{i=1}^{n} p_i$,

$p_{\max} = \max_i p_i$,

$A_i = \sum_{j=1}^{i} p_j$ for $i = 1, 2, \ldots, n$ and

$f_i(t, x, id)$ = the makespan of an optimal schedule with

- total processing time on m_1 of jobs in $V_3 \bigcap \{j_1, j_2, \ldots, j_i\}$ equal to t_3, of jobs in $V_1 \bigcap \{j_1, j_2, \ldots, j_i\}$ equal to t_1, of jobs in $V_4 \bigcap \{j_1, j_2, \ldots, j_i\}$ equal to t_4;
- total processing time on m_2 of jobs in $V_4 \bigcap \{j_1, j_2, \ldots, j_i\}$ equal to t'_4, of jobs in $V_3 \bigcap \{j_1, j_2, \ldots, j_i\}$ equal to t'_3;
- idle time on m_1 is id.

For an easy reference to the above notation see Figure 5.4. Observe that if we know the values $t_1,\ t_3,\ t_4,\ t'_3,\ t'_4$ then we can infer that the total processing time of jobs in $V_2 \bigcap \{j_1, j_2, \ldots, j_i\}$ on m_2 is equal to $V_2 \bigcap \{j_1, j_2, \ldots, j_i\}$ on m_2 is equal to $A_i - (t_3 + t_1 + t_4 + t'_4 + t'_3)$, and thus the makespan of the set of jobs $\{j_1, j_2, \ldots, j_i\}$ can be completely specified by knowing the values of the six parameters that appear as arguments in the function $f_i(\cdot)$.

By definition, the variables $t_1,\ t_3,\ t_4,\ t'_3,\ t'_4$ can assume the following values:

$$0 \leq t_3,\ t'_4 \leq \sum_{i=1}^{n} p_{i1},$$

$$0 \leq t'_3,\ t_4 \leq \sum_{i=1}^{n} p_{i2},\ \textit{and}$$

$$t_1 \leq MS = \sum_{i=1}^{n} p_i.$$

Boundary Conditions: $f_0(0, 0, id, 0, 0) = id$ for $0 \leq id \leq MS$, and

$f_0(t_3, t_1, id, t_4, t'_4, t'_3) = \infty$ otherwise.

Recurrence Relation:

$f_i(t_3, t_1, id, t_4, t'_4, t'_3) =$

$$= \min \begin{cases} \max\{t_3 + t_1 + id + t_4, A_i - (t_3 + t_1 + t_4), f_{i-1}(t_3 - p_{i1}, t_1, id, t_4, t'_3 - p_{i2})\} \\ \qquad if\ \ t_3 \leq A_i - (t_1 + t_3 + t_4) - p_{i2} \\ \max\{t_3 + t_1 + id + t_4, A_i - (t_3 + t_1 + t_4), f_{i-1}(t_3, t_1, id, t_4 - p_{i2}, t'_4 - p_{i1}, t'_3)\} \\ \qquad if\ \ t'_4 \leq t_3 + t_1 + t_4 + id - p_{i2} \\ \max\{t_3 + t_1 + id + t_4, A_i - (t_3 + t_1 + t_4), f_{i-1}(t_3, t_1 - p_i, id + p_i, t_4, t'_4, t'_3)\} \\ \max\{t_3 + t_1 + id + t_4, A_i - (t_3 + t_1 + t_4), f_{i-1}(t_3, t_1, id, t_4, t'_4, t'_3)\}. \end{cases} \quad (1)$$

Optimal Solution:

Let $f_n^\star$ denote the optimal solution to the dynamic program (1) with the above given boundary conditions. Then

$$f_n^\star = \min\{\min\{f_n(t_3, t_1, id, t_4, t_4', t_3') : 0 \le t_3, t_1, t_4, t_4', t_3' \le MS,\ 0 \le id \le p_{\max}\}. \quad (2)$$

Justification:

At the i-th stage of the dynamic program, job j_i has to be scheduled. The first branch of the recursive relationship (1) corresponds to the case that j_i is scheduled to be processed in V_3, the second to the case that j_i is included in V_4, the third to the case that j_i is included in V_1, and the fourth branch corresponds to the case that j_i is included in V_2. For the first and the second branch we have to ensure feasibility of the schedule, i.e. the starting time of task $(i,2)$ must be no earlier than the finishing time of task $(i,1)$. For the first branch this is expressed in the condition $t_3 \le A_i - (t_1 + t_3 + t_4) - p_{i2}$, and For the second branch this is expressed in the condition $t_4' \le t_1 + t_3 + t_4 + id - p_{i2}$. The minimum over the values obtained by the four branches of our recurrence relationship determines the value of $f_i(\cdot)$ as depicted in (1).

For the special case of the 2FFS the recursive relationship (1) can be simplified into

$$f_i(t_3, t_1, id) =$$

$$= \min \begin{cases} \max\{t_3 + t_1 + id, A_i - (t_3 + t_1), f_{i-1}(t_3 - p_{i1}, t_1, id)\} \\ \quad if\ \ t_3 \le A_i - (t_3 + t_1) - p_{i2} \\ \max\{t_3 + t_1 + id, A_i - (t_3 + t_1), f_{i-1}(t_3, t_1 - p_i, id + p_i)\} \\ \max\{t_3 + t_1 + id, A_i - (t_3 + t_1), f_{i-1}(t_3, t_1, id)\}. \end{cases} \quad (3)$$

and the boundary conditions are

$f_0(0,0,id) = id$ for $id = 0, 1, \ldots, \sum_{i=1}^{n} p_i$, and

$f_0(t,x,id) = \infty$ otherwise.

Complexity of the DP algorithm:

Let us refer to the recursive relationship (1). It is evident that none of the variables $t_3, t_1, id, t_4, t'_4, t'_3$ can take on a value greater than MS. Also the index i can take on values $1, 2, \ldots, n$. Hence the number of states of the dynamic program is $\mathcal{O}(nMS^6)$. For each state it takes $\mathcal{O}(1)$ effort to evaluate relationship (1), and therefore the asymptotic complexity of our dynamic programming algorithm is $\mathcal{O}(nMS^6)$. For the special case of 2FFS the recursive relationship (3) has only $\mathcal{O}(nMS^3)$ states and using a similar argument as above we can conclude that the asymptotic complexity of our dynamic programming algorithm for this case is $\mathcal{O}(nMS^3)$.

5.4 Flexible Job Shop versus Traditional Flowshops and Parallel Machines

Johnson's algorithm (JA) can be thought of as a heuristic procedure for 2FJS and 2FFS. Below we perform a worst case analysis of the algorithm when applied to the 2FJS and 2FFS environments. Let C_{FS}, C_{2FFS}, C_{2FJS} be the optimal finishing time of the traditional flowshop, flexible flowshop and flexible job shop respectively. Then we have the following theorem.

Theorem 10 $\frac{C_{FS}}{C_{2FFS}} \leq \frac{C_{FS}}{C_{2FJS}} \leq 2$ *and the bounds are tight.*

Proof. The 2FFS environment is a subsystem of 2FJS and therefore $C_{2FJS} \leq C_{2FFS}$. This proves the first inequality of the theorem. For the second inequality, consider the 2FJS schedule depicted in Figure 5.4. We will show that $C_{FS} \leq 2C_{2FJS}$. Indeed, let S be the sequence of jobs where jobs in the class V_1 are followed by the jobs in the classes V_2, V_3, V_4 without any idle time in between. If a single machine where to process the entire processing load of $p_i = p + i1 + p_{i2}$ of job j_i, it would finish all jobs at time MS. Since a traditional flowshop processes p_{i1} units at stage 1 and p_{i2} units at stage 2, its makespan is no more than MS, i.e. $C_{FS} \leq MS$.

Note that $C_{2FJS} \geq \frac{1}{2}MS$ because a total of MS processing units needs an average of $\frac{1}{2}MS$ units in each of the two machines of 2FJS. The last two relations establish

that $C_{FS} \leq 2C_{2FJS}$. To see that the bounds of the theorem are tight, consider the problem with $n+1$ jobs and processing times $p_{i1} = 1$ for $i = 1, 2, \ldots, n$, $p_{n+1,1} = n$ and $p_{i2} = 0$ for $i = 1, 2, \ldots, n+1$. One can easily see that $C_{2FFS} = C_{2FJS} = n$ while $C_{FS} = 2n$. Thus for this example $\frac{C_{FS}}{C_{2FFS}} = \frac{C_{FS}}{C_{2FJS}} = 2$. □

In effect, the above theorem shows that Johnson's algorithm provides a heuristic for 2FFS and 2FJS with a tight error bound of 2. Observe that the example used to prove tightness of the above bound does not utilize the second stage of FS. Equivalently, the load ratio $LR = \frac{\sum_i p_{i1}}{\sum_i p_{i2}}$ approaches ∞. In practice however LR does not assume such extreme values and often LR is close to 1. Intuitively, when LR is in the neighborhood of 1 the FS environment performs very well (as compared with 2FJS) because both stages have about the same load and therefore it is very likely that both machines work simultaneously. As LR becomes bigger than 1, it harder to keep both stages of the FS simultaneously busy and therefore its performance becomes significantly inferior to 2FJS.

We now turn our attention to the performance of the two parallel identical machine environment (n) as compared with the performance of 2FJS. Let C_n be the optimal makespan of n. The following theorem gives a bound for the ratio $\frac{C_n}{C_{2FJS}}$.

Theorem 11 $\frac{C_n}{C_{2FJS}} \leq 1 + \frac{p_{\max}}{MS}$ *where* $p_{\max} = \max_i p_i$, *and this bound is tight.*

Proof. Consider an optimal schedule S for n. Let C_1, C_2 be the makespan of m_1, m_2 respectively. Assume without loss of generality that $C_1 \leq C_2$ (and thus $C_n = C_2$). Then the job j_l scheduled last on m_2 has processing time $p_l \geq C_2 - C_1$ otherwise we can schedule j_l on m_1 and improve the makespan of n; contradiction to optimality of S. Thus, $C_n \leq C_1 + p_{\max}$. Then, $C_n - \frac{1}{2}MS \leq C_1 + p_{\max} - \frac{1}{2}(C_1 + C_n = p_{\max} - \frac{1}{2}(C_n - C_1)$ $\leq \frac{1}{2}p_{\max}$, or $C_n - \frac{1}{2}MS \leq \frac{1}{2}p_{\max}$. Also, $C_{2FJS} \geq \frac{1}{2}p_{\max}$. Combining the last two relations, we get $\frac{C_n}{C_{2FJS}} \leq 1 + \frac{p_{\max}}{2C_{2FJS}}$. $\leq 1 + \frac{p_{\max}}{MS}$.

To see that this bound is tight consider a problem with n jobs where n is an odd

integer. Also assume that all n jobs have the same processing time t and that there exists at least one job j_i with $p_{i1} = p_{i2} = \frac{t}{2}$. It easy to see that for this problem $C_n = \lceil \frac{n}{2} \rceil t$ while $C_{2FJS} = \frac{nt}{2}$. If $n = 2k + 1$ for some integer k, we have that $\frac{C_n}{C_{2FJS}} \leq 1 + \frac{\frac{1}{2}t}{kt + \frac{1}{2}t} = 1 + \frac{t}{2kt+t}$. Since $p_{\max} = t$ and $MS = (2k + 1)t$, the bound is tight. This completes the proof of the theorem. □

The above theorem indicates that the difference in performance of n and 2FJS depend on $p_{\max}$ and the total workload MS. For big workloads that consist of jobs with small processing time variation, the term $\frac{p_{\max}}{MS}$ of the above bound is small and therefore n and 2FJS have similar performance. For small workloads that consist of a few big jobs the term $\frac{p_{\max}}{MS}$ is big and therefore 2FJS significantly outperforms n.

CHAPTER 6

MANPOWER PLANNING IN MIXED MODEL TRANSFER LINES

6.1 Introduction

Several cells arranged in series and typically connected by a continuous material handling system form a *serial assembly line*. Such lines are designed to assemble component parts and perform any related operation necessary to produce a finished product. Group technology and serial assembly systems can be combined to produce families of parts more economically than traditional process and product layouts. In serial assembly systems the equipment to make similar parts or families of parts is grouped together in a cell designated for these parts. This way a process layout, characteristic of job shops, is changed to a small well-defined product layout. Serial assembly systems can be configured in many ways including a U shape as the Japanese have demonstrated, in an L or in a C shape, see Adam and Ebert (1992). These layouts in conjunction with just-in-time (JIT) production can transform the classic job shop into a world-class manufacturing system.

Among the advantages of serial assembly systems are lower work-in-process inventories, reduced material handling costs, shorter production flow times, simplified production planning of materials and labor, improved visual control and fewer tooling changes. Overall performance often increases by lowering production costs and improving on-time delivery. Quality should increase as well although this might take other interventions beyond the layout change. Several researchers have considered the effect of serial assembly lines in manufacturing including Chen and Adam (1991), Flynn and Jacobs (1987), Fry *et al.* (1987), Goodrich (1988), Hyer and Lea (1984).

Transfer lines are a popular class of serial assembly systems for which the handling system indexes parts on the line ahead to the next cell every c time units. The time period c is called the production cycle of the transfer line. In a transfer line consideration of breakdowns is of great importance (see Askin and Standridge, 1993 pp 67). The manufacturing environment of the problem treated in this chapter is a transfer line. The automobile, fire truck, aircraft and PC board assembly are just a few applications of labor intensive transfer lines. In this chapter we address manpower planning issues in such lines.

Before we discuss the relation of our model with existing literature, we will first define it precisely. A set $J = \{J_1, J_2, \ldots, J_n\}$ of jobs is to be processed on a serial assembly line that consists of m stations. Each job J_i consists of m ordered tasks. The j-th task must be processed by the j-th station ST_j of the assembly line. Every task must be completed within a prespecified production cycle of c periods. We assume that the number of workers needed to complete the j-th task on ST_j within c periods is M_{ij}. Hence, each job J_i is an m-tuple of manpower requirements. Our problem is to find a permutation of the jobs that minimizes the maximum manpower requirement over all production cycles. To compute the manpower requirement during some cycle one has to add up the individual requirements of all the stations of the assembly line. We assume that workers are assigned to stations only at the beginning of each production cycle and each worker is eligible to be assigned to every station.

The transfer line environment that we study bears some similarities with the mixed model assembly line environment. Due to market considerations recent attention to product variety has led to the use of *mixed model lines* where several variations of a basic stable design are allowed to be on the line at the same time. Example of such products could be different models of a car where each model may come with a variety of options. Then, depending on the model and options each car has different processing requirements. In most mixed model assembly lines the conveyance

system moves at constant speed and the models are introduced in the line in equal time intervals, see Dar-El and Cother (1975), Okamura and Yamashima (1979), Thomopoulos (1967). Safety buffers are utilized to hedge against starvation and blockage of stations.

Mixed model lines have received significant attention over the last years. A great part of the literature addresses design issues that include the number of stations as well as the throughput time of the line (see Bard, 1989 and Bard *et al.*, 1992). In a survey paper Baybars (1986) presents a detailed description of the assembly line balancing problem and summarizes findings for exact algorithms. Baybars and Frieze (1986) discuss the performance of heuristics. Another issue encountered in mixed model assembly lines is the sequencing of models with primary objectives the control of work-in-process inventories and smooth production. Research in this area includes the work of Dar-El and Cother (1975), Miltenburg (1989), Okamura and Yamashina (1979), Thomopoulos (1970), Steiner and Yeomans (1993) and Wittrock (1985). Other issues that arise in mixed model lines have been detailed by Dar-El (1977).

In the mixed model assembly line the usual assumption is that workers move in an *asynchronous fashion* i.e. a worker moves into a bottleneck station as soon as he completes his current task. As a result the time to subassemble each model in each station is not constant and depends on the labor movement along the line. Note that this assumption removes the manpower problem since in this case the workers' idle time is minimized. However, to ensure smooth production the above assumption rely on intelligent inventory control and many times on the installation of buffers.

The difference between our problem and the mixed model assembly line lies in the assumptions made for the movement of jobs and workers along the line. More specifically we assume a common production cycle for all tasks and synchronous worker

movement. In what follows we further discuss these two assumptions. In some automobile, fire truck and aircraft assembly safety buffers are impossible to maintain (e.g. it is not cost justified to keep in inventory fire truck engines). In this case the common production cycle assumption of our model is a good way to hedge against line breakdown. This assumption allows the coordination of the supply chain so that every major expensive part used during a subassembly is delivered to the appropriate station just before the commencement of the next production cycle. In this case materials requirement planning can be carried out with great accuracy and the manufacturer is able to request parts from his vendor well in advance. From the managerial point of view this coordination is of unique importance because it eliminates major investments in work-in-process inventories and reduces the risk of stopping the assembly line which can be extremely costly for labor intensive operations.

Our experience with manufacturers that rely heavily on labor is that managing asynchronous worker movement is an extremely difficult managerial task. Asynchronous movement allows workers to 'hide' in the crowd since it is not easy to keep track of the exact station that each worker should be at every point in time. As a result a great amount of working capacity is lost and before long the entire line is in havoc. Then, a simple approach to manpower management is to assign as few workers in each station as necessary to complete the subassembly of the corresponding station within the c periods of the production cycle. Workers are assigned to a station in the beginning of a cycle and do not leave this station before the cycle has expired. We refer to this approach as *synchronous worker movement.*

The synchronous worker movement simplifies the manpower planning by limiting management intervention regarding communication and coordination of the workforce to the beginning of the production cycle. Also it makes easier to recognize problems on the line since at any time a number of workers are directly responsible for their station.

It is evident that under the synchronous worker movement assumption some stations might be assigned slightly greater working capacity than they actually need to complete a subassembly in c periods. In reality however, the workers of each station pace themselves while making sure that their subassembly will be complete at the end of the production cycle. Also, observe that depending on the various models currently processed, the line will have less than maximum manpower requirements during some production cycles. In practice the workforce consists of permanent employees and temporary workers that are hired from a pull of contracted labor. In case that the permanent workforce exceeds the requirements of a production cycle the extra workers are assigned to ancillary operations, preparation work for upcoming models or for training in other stations in order to retain high skill flexibility. When the permanent workforce is insufficient, extra workers are temporarily hired.

Note that the common production cycle c can be easily adjusted by adding or removing workers. Therefore, the tools that we develop in the following sections can be used not only for configuring the assembly line but also for testing the effects of design changes on the manufacturing process. For example, redesigning a fender changes the labor requirements associated with a particular model. Such changes are especially important in the rapidly innovative industries mentioned earlier. For this reason we are interested in developing fast algorithms to solve our manpower problem so that it can be used as a design tool.

Some work has been done on manpower planning of assembly lines that produce a single item. Bartholdi (1992) has presented a case study in which workers perform on different tasks of the same item while Pinto *et al.* (1981) have developed branch and bound and heuristic procedures for the case that workers perform the same task on different items. The importance of decentralized workforce control is captured by Bartholdi and Eisenstein (1993) where every worker of the assembly line follows a simple rule of what to do next. To the best of our knowledge no research has been

done on the sequencing problem of mixed model transfer lines with primary objective to minimize manpower. The protocol of operations described above is currently implemented by a fire truck manufacturer from which this research is motivated. In this fire engine plant all stations have the same production cycle and the manpower is assigned in the beginning of each production cycle.

The rest of the chapter is divided into 7 sections. In Section 6.2 we define the problem, and in Section 6.3 we solve optimally the case of 2-station assembly lines. We show that the problem becomes $\mathcal{NP}$-complete for three stations, in Section 6.4. This observation leads to the search for heuristic algorithms. In Section 6.5 we develop several heuristics as well as lower bounds to evaluate them. We analyze the worst case error bound and average performance of these heuristics in Section 6.6. In Section 6.7 we exploit the trade-off between the length of the production cycle and the corresponding manpower requirements. Also we offer a way to divide the workforce in workers that are specialized for some station, and utility workers. Closing remarks are made in Section 6.8.

6.2 Description of the Assembly Line Problem

The following notation will be used throughout this chapter:

n: number of jobs

m: number of stations

c: production cycle

J_i: the i-th job of $J = \{J_1, J_2, \ldots, J_n\}$

ST_j: the j-th station of $ST = \{ST_1, ST_2, \ldots, ST_m\}$

M_{ij}: the manpower requirement of job J_i on station ST_j in order to complete the j-th task of J_i in c periods, $i = 1, 2, \ldots, n$, $j = 1, 2, \ldots, m$.

We consider a serial assembly line with m stations. A target production cycle of c periods is given along with a set J of n jobs. Each job $J_i \in J$ has an associated

m-tuple where the j-th component M_{ij}, is the number of workers needed to process job J_i through the j-th station of the assembly line within c time periods. All jobs enter from the first station and are transferred to the next station of the line after c periods. The objective is to find a permutation of the jobs that minimizes the maximum workforce level of the production cycles.

For ease of presentation we give the formulation of the problem when $m = 2$. Then we have the 2-station transfer line problem which we will denote by *2SAL*. This problem is stated as follows: A set J of simultaneously available jobs is given to be processed on two stations. Every job J_i consists of two tasks. The first task must be processed at ST_1 for c periods and upon completion, the second task continues at ST_2 where it stays for another c periods. For convenience, we will use the pair (M_{i1}, M_{i2}) to denote job J_i. It is assumed that no station can handle more than one task at a time and no task can be interrupted once it has begun processing. Let $S = J_{[1]}J_{[2]}\ldots J_{[n]}$ denote a sequence of jobs. The manpower needed for S, denoted by $M(S)$, is

$$M(S) = \max\{M_{J_{[1]},1}, M_{J_{[n]},2}, \max_{1\leq i\leq n-1}(M_{J_{[i]},2} + M_{J_{[i+1]},1})\}$$

The problem is to find an optimal schedule S^* such that $M(S^*) \leq M(S)$ for every schedule S. The definition of the problem is similar for more than 2 stations.

To facilitate the rest of the chapter we introduce an example.

Example 1: Consider the 2-station, 5-job problem of Table 6.1.

The sequence $S = J_3J_4J_5J_2J_1$ is also written down in Table 6.1. In the layout used each of the 6 columns correspond to one of the $n + m - 1 = 6$ production cycles. For example, during the 2nd production cycle, ST_2 requires 1 worker while ST_1 requires 8. Namely, job J_3 occupies ST_2 at the same production cycle that J_4 occupies ST_1. For this particular sequence S, we observe that $M(S) = 13$ and this level is attained in the third and fourth production cycle.

Table 6.1. A 5-job example for 2SAL

Job	Manpower Req's	
	M_{i1}	M_{i2}
J_1	2	7
J_2	4	5
J_3	6	1
J_4	8	3
J_5	10	9

6 1 ← J_3
8 3 ← J_4
10 9 ← J_5
4 5 ← J_2
2 7 ← J_1

6.3 The 2SAL Problem

In this section, we will provide a polynomial algorithm that solves the 2SAL problem optimally. Our approach is based on a representation of the problem by a bipartite graph.

6.3.1 Formulation of 2SAL

A dummy job $(0,0)$ is added to the job set, whose cardinality now becomes $n+1$. Let $B= B(E_1, E_2)$ be a $(n+1) \times (n+1)$ complete bipartite graph. The set E_1 corresponds to the manpower requirements of the first task of jobs in J. The vertices $\{v_0, v_1, v_2, \ldots, v_n\}$ in E_1 are labeled by $M_{01} = 0, M_{11}, M_{21}, \ldots, M_{n1}$. We assume that $M_{01} \leq M_{11} \leq M_{21} \leq \ldots \leq M_{n1}$ and that the label of v_i is M_{i1} for $0 \leq i \leq n$. The set E_2 corresponds to the manpower requirements of jobs in J at ST_2. The vertices $\{u_0, u_1, u_2, \ldots, u_n\}$ in E_2 are labeled by $M_{02} = 0, M_{12}, M_{22}, \ldots, M_{n2}$. We assume that $M_{02} \leq M_{12} \leq M_{22} \leq \ldots \leq M_{n2}$ and that the label of u_i is M_{i2} for $0 \leq i \leq n$.

Let (M_{i1}, M_{j2}) be a job in J for some $0 \leq i, j \leq n$ with corresponding vertices v_i, u_j. Then link v_i and u_j by an edge. This results to a set of $n+1$ edges which

correspond to the $n+1$ jobs. Let I be this set of edges. Clearly I is a matching of B because no two edges have a vertex in common. Let $S = J_{[1]}J_{[2]}\ldots J_{[n]}$ be a job sequence for 2SAL. Then S induces in B, a second matching N, as follows: For $1 \leq i \leq n-1$ add the edge $(M_{J_{[i]},2}, M_{J_{[i+1]},1})$ in N. Also, include the edges $(M_{J_{[n]},2}, M_{01})$ and $(M_{02}, M_{J_{[1]},1})$ in N. To each edge (v_i, u_j) in N we associate a weight w_{ij} equal to the sum of the labels corresponding to the endpoints of that edge i.e. $w_{ij} = M_{i1} + M_{j2}$. We illustrate the construction of the bipartite graph B by an example.

Example 2: Consider the instance of Example 1 in Section 6.2. Then the six jobs (including the dummy job J_0) can be expressed as pairs by $J_0 = (0,0)$, $J_1 = (2,7)$, $J_2 = (4,5)$, $J_3 = (6,1)$, $J_4 = (8,3)$ and $J_5 = (10,9)$. Then, I is the matching formed by these pairs. We depict B so that the labels M_{i1} appear in nondecreasing order while labels M_{j2} appear in nonincreasing order; see Figure 6.1.

Given the sequence $S = J_3J_4J_5J_2J_1$ the matching N_1 (corresponding to the dotted edges of Figure 6.1a) is $N_1 = \{(0,7),(2,5),(4,9),(6,0),(8,1),(10,3)\}$ and $M(S) = 13$. On the other hand the matching $N_2 = \{(0,3),(2,9),(4,7),(6,5),(8,0),(10,1)\}$ does not correspond to a sequence because $I \bigcup N_2$ forms two cycles (see Figure 6.1b).

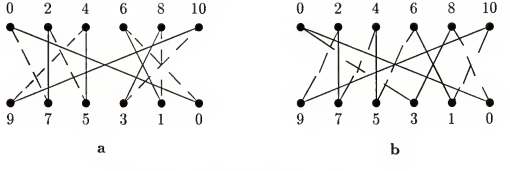


Figure 6.1. The matchings I and N

Proposition 3 *A matching N determines a job sequence for 2SAL iff $C = I \bigcup N$ is a unique cycle.*

Proof. Suppose that N determines a job sequence $J_{[1]} \ldots J_{[n]} J_0$. Then, $C = I \bigcup N$ is the graph that consists of the edge $(M_{J_{[1]},1}, M_{J_{[1]},2}) \in I$ corresponding to $J_{[1]}$ followed by $(M_{J_{[1]},2}, M_{J_{[2]},1}) \in N$ followed by the edge $(M_{J_{[2]},1}, M_{J_{[2]},2}) \in I$ corresponding to $J_{[2]} \ldots$ followed by $(M_{J_{[0]},2}, M_{J_{[1]},1}) \in N$. Clearly, C is a unique cycle that includes all edges of I, N.

On the other hand, assume that there is a matching N such that $C = I \bigcup N$ is a cycle on $2(n+1)$ vertices. Then, since I is the matching corresponding to the set of jobs, N must be a matching whose edges alternate with the edges of I in C. Also, since C is a cycle on $2(n+1)$ vertices, each of I, N includes $n+1$ edges. From C we can construct a job sequence for 2SAL as follows. Define an orientation for C from M_{i1} to M_{i2} where J_i is an arbitrary job. Starting from J_0 scan C according to the above orientation. Let $J_{[1]} \ldots J_{[n]}$ be the order in which the edges of I are scanned. Since C is a unique cycle every job $J_i \in J$ is included in the above sequence precisely once; i.e. N determines a job sequence. □

Note that if $I \bigcup N$ is not a cycle then N does not correspond to a job sequence. Proposition 3 shows that the 2SAL problem is a Bottleneck Matching problem (see Papadimitriou and Steiglitz, 1982 pp 245) with the additional constraint that $I \bigcup N$ forms a unique cycle. The bipartite graph formulation of 2SAL will be explored further to solve optimally the 2SAL problem in polynomial time.

6.3.2 Solution Procedure for 2SAL

Let I be the matching corresponding to the jobs, as described in the previous subsection. We want to find a matching $N \neq I$ such that $I \bigcup N$ is a cycle and such that N minimizes the maximum weight on its edges. Recall that $0 \leq M_{11} \leq M_{21} \leq \ldots \leq M_{n1}$ are the labels of the vertices $v_0, v_1, \ldots, v_n$ of the bipartite graph B and $0 \leq M_{12} \leq M_{22} \leq \ldots \leq M_{n2}$ are the labels of the vertices $u_0, u_1, \ldots, u_n$ respectively.

Now we develop the algorithm O-2SAL which as we will show shortly produces an optimal solution for 2SAL. First, we observe the following property.

Property 9 *Let* M^* *be the optimal solution for 2SAL. Then,* $M^* \geq \max\{M_{i1} + M_{n-i,2},\ 0 \leq i \leq n\}$

Property 9 follows directly from the following fact. In order to match the elements of two n-element sequences so that the maximum sum of matched elements is minimized, then the i-th element of the nondecreasing order of the first sequence should be matched with the i-th element of the nonincreasing order of the second sequence. For a formal proof of this property see Hsu (1984).

Property 9 signifies that the manpower requirement of $M_{i1} + M_{n-i,2}$ corresponding to the edge $(M_{i1}, M_{n-i,2})$, does not exceed the optimal level M^*. Hence, a first approach to solving 2SAL is to let N be the set of edges $(M_{i1}, M_{n-i,2}) : 0 \leq i \leq n$. If $I \bigcup N$ forms a unique cycle, then N corresponds to an optimal sequence. This is implemented at step 0 of the algorithm that follows. In general, however, $I \bigcup N$ forms a number of cycles. In this case we have to devise an optimal strategy of merging the various cycles into one. This strategy is outlined at step 1 of algorithm O-2SAL below.

Algorithm O-2SAL

Input : M_{ij} $i = 1, 2, \ldots, n$, $j = 1, 2$ and matching I

Output: Optimal sequence S^* and manpower level M^*

[0] Order $\{M_{i1} : i = 0, 1, 2, \ldots, n\}$, $\{M_{i2} : i = 0, 1, 2, \ldots, n\}$ in nondecreasing order

$N := \emptyset$

Add to N all edges $(M_{i1}, M_{n-i,2})$, $i = 0, 1, 2, \ldots, n$

if $I \bigcup N$ is a unique cycle *then* STOP

[1] Order $\{M_{i1} + M_{n-i+1,2} : i = 1, 2, \ldots, n\}$, in nondecreasing order

i:=0

repeat

i:=i+1

if M_{i1}, $M_{n-i+1,2}$ belong to different cycles *then begin*

Delete the edge (M_{i1}, M_{y2}) from N where M_{y2} is the label of the vertex adjacent to v_i (whose label is M_{i1}) in N

Delete the edge $(M_{x1}, M_{n-i+1,2})$ from N where M_{x1} is the label of the vertex adjacent to u_{n-i+1} (whose label is $M_{n-i+1,1}$)

Add the edges (M_{x1}, M_{y2}) and $(M_{i1}, M_{n-i+1,2})$ to N

endif

until $I \bigcup N$ is a unique cycle

Let k be the number of distinct cycles in $I \bigcup N$. If $k > 1$ step 1 of O-2SAL will merge these cycles (two at each iteration) into a unique cycle. Hence after $k-1$ iterations of step 1 the algorithm will terminate. We will illustrate the algorithm by means of an example.

Example 3: Consider the instance of Example 1 in Section 2. At step 0 we add the edges $(0,9),(2,7),(4,5),(6,3),(8,1),(10,0)$; the dotted edges of Figure 6.2a. At this point, $I \bigcup N$ consists of four different cycles. Hence we go to step 1. The minimum of $\{M_{i1} + M_{n-i+1,2} : i = 1,2,\ldots,n\}$ for which M_{i1}, $M_{n-i+1,2}$ belong to different cycles is $M_{11} + M_{52} = 2 + 9 = 11$. Delete $(0,9)$, $(2,7)$ and add $(2,9),(0,7)$. Then we are left with three cycles. Delete $(4,5)$, $(0,7)$ and add $(4,7),(0,5)$. Now $I \bigcup N$ forms two cycles; namely $0-0\cdots10-9\cdots2-7\cdots4-5\cdots0$ and $6-1\cdots8-3\cdots6$. Delete $(0,5)$, $(6,3)$ and add $(6,5),(0,3)$ to merge the two cycles into one. The cycle induced by O-2SAL is $C = 0-0\cdots10-9\cdots2-7\cdots4-5\cdots6-1\cdots8-3\cdots0$, for which $M(C) = 11$. The optimal sequence is $J_5J_1J_2J_3J_4$ as we see from Figure 6.2b. As mentioned above, at the end of Step 0, $I \bigcup N$ consists of a number of cycles. By

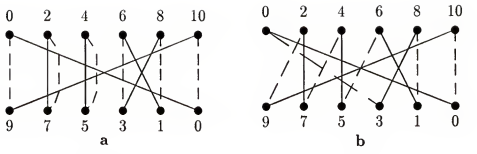


Figure 6.2. An application of 2SAL

construction, every cycle traverses an even number of vertices because once the node M_{i1} is saturated by a cycle, node $M_{n-i,2}$ is saturated by the same cycle, as well.

Let's describe the structure possessed by a cycle C at the end of step 0 of O-2SAL. Observe that every time that two cycles are merged into one, we delete from $I \bigcup N$ two edges and then add another two edges into $I \bigcup N$. Because of this, C consists of several blocks where each block is defined as a set of consecutive (i.e. one next to the other, at the graph B) pairs of vertices v_i, u_{n-i}, belonging to C. Hence, C traverses a block of vertices $M_{i_1,1}, M_{n-i_1,2}, M_{i_1+1,1}, M_{n-i_1-1,2}, \ldots, M_{i_2,1}, M_{n-i_2,2}$ possibly followed by another block of vertices $M_{i_3,1}, M_{n-i_3,2}, M_{i_3+1,1}, M_{n-i_3-1,2}, \ldots, M_{i_4,1}, M_{n-i_4,2}$ and so on, with $0 \leq i_1 \leq i_2 < i_3 \leq i_4 < \ldots < i_{2k-1} \leq i_{2k}$, assuming that C consists of k distinct such blocks. Consider for example the four cycles of Figure 6.2a. The cycle $0 - 0 \cdots 10 - 9 \cdots 0$ contains 4 edges and consists of two blocks. The first block contains the vertices M_{01}, M_{52} while the second contains the vertices M_{51}, M_{02}. The cycle $2 - 7 \cdots 2$ has 2 edges and consists of one block that contains the vertices M_{11}, M_{42}. Similarly, the cycle $6 - 1 \cdots 8 - 3 \cdots 6$ has 4 edges and consists of one block that contains the vertices $M_{31}, M_{41}, M_{12}, M_{22}$.

One can see that at step 1, O-2SAL computes the minimum among $M_{i1} + M_{n-i+1,2}$ where M_{i1} is traversed by C and $M_{n-i+1,2}$ is not traversed by C, or the other way around. This reflects the intuition that the cheapest way to incorporate C into a bigger cycle, is to utilize one of the edges that attain the above minimum, say M. To

prove optimality of O-2SAL it is necessary to show that $M \leq M^*$. This is done in the next theorem.

Theorem 12 *The solution produced by O-2SAL is optimal.*

Proof. If at the end of step 0, $I \bigcup N$ is a unique cycle, then we saw that Property 9 guarantees optimality. Else we proceed to step 1. Let C be a cycle produced by O-2SAL at step 0. We saw that C consists of a number of blocks of vertices. Without loss of generality. C is comprised by the blocks $M_{i_1,1}, M_{n-i_1,2}, M_{i_1+1,1}, M_{n-i_1-1,2}, \ldots, M_{i_2,1}$, $M_{n-i_2,2}, \ldots, M_{i_{2k-1},1}, M_{n-i_{2k-1},2}, M_{i_{2k-1}+1,1}, M_{n-i_{2k-1}-1,2}, \ldots, M_{i_{2k},1}, M_{n-i_{2k},2}$ with $0 \leq i_1 \leq i_2 < \ldots < i_{2k-1} \leq i_{2k}$.

To prove the theorem we need to show that at every iteration of step 1, the two added edges (M_{x1}, M_{y2}) and $(M_{i1}, M_{n-i+1,2})$ are such that $M_{x1} + M_{y2} \leq M^*$ and $M_{i1} + M_{n-i+1,2} \leq M^*$.

Consider the edge (M_{x1}, M_{y2}). If $y \leq n-x$ then $M_{x1}+M_{y2} \leq M_{x1}+M_{n-x,2} \leq M^*$ from Property 9. The case $y > n-x$ never occurs in our algorithm (except of course for the edge $M_{i1} + M_{n-i+1,2}$ which will be discussed next), because we only add edges (M_{x1}, M_{y2}) with $y < n-x$ at step 1 or $y = n-x$ at step 0.

Hence we are left to show that $M_{i1}+M_{n-i+1,2} \leq M^*$. Consider an optimal solution S^* for 2SAL and $I \bigcup N^*$ the corresponding cycle. Since the vertices traversed by C are also traversed by $I \bigcup N^*$, there must be at least one edge $e = (M_{r1}, M_{s2})$ in N^* with one endpoint in C and the other not in C. We distinguish the following cases for M_{r1}, M_{s2}:

Case i: $s > n-r$, $M_{s2} \in C$, $M_{r1} \notin C$.

The bipartite graph for this combination looks as in Figure 6.3a. Suppose that M_{s2} belongs to the l-th block of vertices of C. Let this block be $M_{i_{2l-1},1}, M_{n-i_{2l-1},2}, \ldots, M_{i_{2l},1}$, $M_{n-i_{2l},2}$ with $i_{2l-1} \leq i_{2l}$. Since $M_{r1} \notin C$ and $s > n-r$, we have that $M_{r1} \geq M_{i_{2l}+1,1}$. Also, since $s \geq n - i_{2l}$ we have that $M_{s2} \geq M_{n-i_{2l},2}$. Therefore, $M_{n-i_{2l},2} + M_{i_{2l}+1,1} \leq$

$M_{s2} + M_{r1} \leq M^*$. Hence, $\min_{M_{n-i+1,2} \in C, M_{i1} \notin C}\{M_{i,1} + M_{n-i+1,2}\} \leq M^*$.

Case ii: $s > n - r$, $M_{s2} \notin C$, $M_{r1} \in C$.

This combination is depicted in Figure 6.3b and is symmetric to Case i. From this case we obtain $\min_{M_{i1} \in C, M_{n-i+1,2} \notin C}\{M_{i1} + M_{n-i+1,2}\} \leq M^*$. *Case iii:* $s < n - r$, $M_{s2} \in C$,

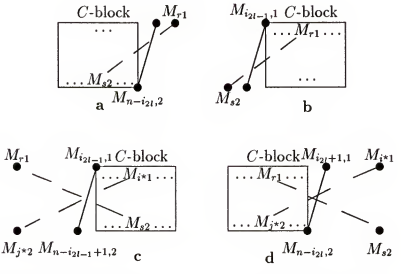


Figure 6.3. Supporting figure for the proof of Theorem 12

$M_{r1} \notin C$. This combination is depicted in figure 3c. Consider the block of vertices of C that includes M_{s2}. Let this block be $M_{i_{2l-1},1}, M_{n-i_{2l-1},2}, \ldots, M_{i_{2l},1}, M_{n-i_{2l},2}$ with $i_{2l-1} \leq i_{2l}$. Then, in N^* the $n - i_{2l-1} + 1$ vertices $M_{i_{2l-1},1}, \ldots, M_{n1}$ are incident to $n - i_{2l-1} + 1$ vertices from $M_{02}, \ldots, M_{n2}$. Since there are only $n - i_{2l-1}$ available vertices of the form M_{j2} with $j \leq n - i_{2l-1}$, there must be a vertex $M_{i^*,1}$ with $i^* \geq i_{2l-1}$ incident to some $M_{j^*,2}$ with $j^* \geq n - i_{2l-1} + 1$. Then, $M_{i_{2l-1},1} + M_{n-i_{2l-1}+1,2} \leq M_{i^*,1} + M_{j^*,2} \leq M^*$ and therefore $\min_{M_{i1} \in C, M_{n-i+1,2} \notin C}\{M_{i1} + M_{n-i+1,2}\} \leq M^*$.

Case iv: $s < n - r$, $M_{s2} \notin C$, $M_{r1} \in C$. This combination is depicted in Figure 6.3d and is symmetric to Case iii. From this case we obtain that $\min_{M_{i1} \notin C, M_{n-i+1,2} \in C}\{M_{i1} + M_{n-i+1,2}\} \leq M^*$.

Therefore, $\min_i\{M_{i1} + M_{n-i+1,2} : M_{i1}, M_{n-i+1,2}$ *belong to different cycles*$\} \leq M^*$. Since the four cases exhaust all possible combinations for M_{r1} and M_{s2}, the

theorem is proved. □

Complexity of O-2SAL

At step 0 we can store the vertices of each of the various cycles in a linked list in $\mathcal{O}(n)$ time. N can be stored in two vectors A, B so that $A[i]$, $B[i]$ are the vertices adjacent to v_i, u_i respectively. By scanning the vertices of the bipartite graph B from left to right, and by using the vectors A, B, the total effort spent at step 0 for labeling vertices that belong to different cycles, is $\mathcal{O}(n)$ because every vertex needs to be considered once. At step 1, the ordering of edges $(M_{i1}, M_{n-i+1,2})$ with endpoints belonging to different cycles, takes time at most $\mathcal{O}(n \log n)$. Let C be a stack storing the labels M_{i1} top to bottom, in nondecreasing order of $M_{i1} + M_{n-i+1,2}$. Using C, we can detect which two cycles we merge next. This is done by checking whether the endpoints of the edge $(M_{i1}, M_{n-i+1,2})$ corresponding to top element M_{i1} of the stack, belong to different cycles. If they do, the two cycles are merged next. If not, the element at the top of the stack is discarded. Hence the total effort spent for detecting which cycles to merge is $\mathcal{O}(n)$. The additions and deletions of step 1 are done in $\mathcal{O}(1)$ time. Also, merging two cycles is equivalent to merging two linked lists; this is done in $\mathcal{O}(1)$ time as well. Thus the overall complexity of O-2SAL is $\mathcal{O}(n \log n)$ due to the sorting that takes place at steps 0 and 1.

Having proved that the case $m = 2$ is polynomially solvable, we consider the complexity of the problem for three or more stations. This is the focus of the next section.

6.4 Complexity Result

In this section we show that the manpower problem of an assembly line with three stations (3SAL) is strongly $\mathcal{NP}$-complete. We will use a reduction from the Numerical 3-Dimensional Matching problem which is known to be $\mathcal{NP}$-complete in the strong sense; see Garey and Johnson (1979).

Numerical 3-Dimensional Matching (N3DM)

INSTANCE: Given disjoint sets A, B, C each containing n nonnegative numbers, and a bound $\bar{M}$.

QUESTION: Can $A \bigcup B \bigcup C$ be partitioned into n disjoint sets $S_1, S_2, \ldots, S_n$ such that each S_i contains exactly one element from each of A, B, C and such that $\sum_{s \in S_i} s = \bar{M}$ for $i = 1, 2, \ldots, n$?

Without loss of generality we can assume that $\forall\ a \in A,\ 0 < a \leq K$, $\forall\ b \in B,\ K < b \leq 2K$ and $\forall\ c \in C,\ 3K < c \leq 4K$ for some value $K > 0$, otherwise we can add $K := \max\{x : x \in A \bigcup B \bigcup C\}$ to every element in B, $3K$ to every element in C and $3K$ to $\bar{M}$ so that the triplet A', B', C' satisfies the condition, where $A' = A$, $B' = \{K + b : b \in B\}$ and $C' = \{3K + c : c \in C\}$. In the next theorem we assume that the triplet A, B, C already satisfies this extra condition.

Theorem 13 *The 3SAL problem is $\mathcal{NP}$-complete in the strong sense.*

Proof. Let $X = (A, B, C)$ be an instance of N3DM. From X, construct an instance I of 3SAL having the following $5n + 1$ jobs:

$\{(\bar{M}, 1, a) : a \in A\}$

$\{(\bar{M} - 1, \bar{M} - a, a) : a \in A\}$

$\{(0, b, 0) : b \in B\}$

$\{(c, c, 0) : c \in C\}$

$\{(\bar{M} - c, 0, 0) : c \in C\}$

$\{(\bar{M}, \bar{M}, \bar{M})\}$

We will show that X has a solution if and only if I has a solution with manpower requirements $\bar{M}$. Suppose X has a solution. Let (a, b, c) be a triplet of X. We have that $a + b + c = \bar{M}$. To every triplet of X, we correspond the following ordered block of jobs: $(\bar{M}, 1, a), (\bar{M} - 1, \bar{M} - a, a), (0, b, 0), (c, c, 0), (\bar{M} - c, 0, 0)$. Given a solution for X as an ordered set of n triplets, we have an order of the n corresponding blocks,

i.e. an order for $5n$ jobs. Letting job $(\bar{M}, \bar{M}, \bar{M})$ to be the job following the previous $5n$, we have a sequence S for I. We claim that S is optimal. In S, the manpower level is $\bar{M}$ during every cycle. Since the total number of cycles required to finish I is $5n+3$ and the total requirements are $(5n+3)\bar{M}$, the optimum manpower level cannot be less than $\bar{M}$. Therefore S is optimal.

On the other hand suppose there exists a solution for I with maximum manpower requirement $\bar{M}$. Then the manpower level during any cycle must equal $\bar{M}$ because the total number of cycles required to finish I is $5n+3$ and the total requirements are $(5n+3)\bar{M}$. Then, the last job must be $(\bar{M}, \bar{M}, \bar{M})$ since this is the only job that requires $\bar{M}$ workers in the last station. The first job must be $(\bar{M}, 1, a_i)$ for some $a_i \in A$ because this is the only type of job that requires $\bar{M}$ workers in the first station, besides $(\bar{M}, \bar{M}, \bar{M})$. The second job must be $(\bar{M}-1, \bar{M}-a_j, a_j)$ for some $a_j \in A$ because this is the only set of jobs that requires $\bar{M}-1$ workers in the first station and thus adds up to $\bar{M}$ units with the second task of the first job. We will show that in order for the manpower level to be $\bar{M}$ during the third cycle, we must have $a_i = a_j$. Indeed, let x be the manpower requirement of ST_1 during the third cycle. Then the total requirement for this cycle is $a_i + (\bar{M}-a_j) + x$ and this must equal $\bar{M}$, i.e. $a_i - a_j + x = 0$ or $x = a_j - a_i < K$. But the only set of jobs that has requirement less than K on ST_1 is $\{(0, b, 0) : b \in B\}$ in which case $x = 0 \Rightarrow a_i = a_j = a$.

The next job must be from the set $(0, b, 0)$ for some $b \in B$ because this is the only job that requires 0 units in the first station and thus adds up to $\bar{M}$ units with the first two jobs. The next job could be from either set $\{(\bar{M}-c, 0, 0) : c \in C\}$ or $\{(c, c, 0) : c \in C\}$. If it is from the former, then there must exist (a, b, c) such that $a + b + (\bar{M}-c) = \bar{M}$. But then, $a + b = c$ which contradicts the fact that $a + b \leq 3K < c$, according to the definition of a, b, c. Thus a job $(c, c, 0)$ for some $c \in C$ is the only choice. Then, the next job must be $(\bar{M}-c, c, 0)$ because this is the only job that adds up to $\bar{M}$ workers with the previous two jobs.

Repeating the above argument we see that an optimal sequence consists of the previous block repeated n times with last job being $(\bar{M}, \bar{M}, \bar{M})$. Since in every cycle the manpower level must be $\bar{M}$, within each block it must be true that $a+b+c = \bar{M}$. Naturally, if there exists a sequence S for I with $M(S) = \bar{M}$, this sequence induces a solution to X by each triplet been the triplet (a, b, c) inside each block. This completes the proof. □

The following questions arise naturally. What determines the difficulty of scheduling an m-station assembly line optimally? Can this difficulty be accounted by Group Technology (GT) techniques in the design of the line? Note that if $n \geq m$ then for every sequence S we have

$$\sum_j \min_i M_{ij} \leq M(S) \leq \sum_j \max_i M_{ij}$$

and therefore

$$\frac{M(S)}{M^*} \leq \frac{\sum_j \max_i M_{ij}}{\sum_j \min_i M_{ij}} \leq \frac{\max_{i,j} M_{ij}}{\min_{i,j} M_{ij}}.$$

If for instance $\max_{i,j} M_{ij} \leq 2 \min_{i,j} M_{ij}$ for every station j, then $\frac{M(S)}{M^*} \leq 2$ for every sequence S. Therefore, GT can improve the manning of transfer lines by balancing out the workload to the various stations or cells. In Section 6.5 we show empirically that the smaller the ratio $\frac{\max_{i,j} M_{ij}}{\min_{i,j} M_{ij}}$, the easier it is to find good heuristic solutions.

Theorem 13 shows that it is very unlikely that there exists an optimal algorithm that solves $mSAL$ in polynomial time. Thus, we focus our attention to heuristic approaches in order to obtain satisfactory solutions for the 3SAL and the general $mSAL$ problem, in reasonable time. To evaluate these heuristics we need to develop tight lower bounds. We focus on these issues in the next section.

6.5 Lower Bounds and Heuristics

Consider the general $mSAL$ problem with m stations. Let M^* denote the optimal value of $mSAL$ and M_H the corresponding value of some heuristic H. The following result provides an upper bound on the performance of any heuristic H.

Theorem 14 For every sequence S, $M(S) \leq mM^*$.

Proof. Suppose that the value $M(S)$ is attained in the production cycle k where $1 \leq k \leq n+m-1$. Then $M(S)$ is the summation of the manpower levels required by the m stations during cycle k. Let these requirements be $a_1, a_2, \ldots, a_m$.

Then $a_r \leq \max_{ij} M_{ij} \leq M^*$ for every $1 \leq r \leq m$, $1 \leq i \leq n$ and $1 \leq j \leq m$. Therefore, $M(S) \leq mM^*$. □

A simple lower bound can be established for $mSAL$ by distributing manpower requirements as even as possible throughout the $n+m-1$ production cycles. Namely,

$$M^* \geq \lceil \frac{\sum_{ij} M_{ij}}{n+m-1} \rceil.$$

Before presenting any heuristic we need to develop tight lower bounds for $mSAL$ in order to be able to compute the deviation of the heuristic solution from the optimal. We do this by utilizing a surrogate relaxation approach (see e.g. Glover 1975), based on the following integer formulation. Given n jobs with manpower requirement vectors $\vec{M}_1, \vec{M}_2, \ldots, \vec{M}_n$, define the 0-1 variable x_{ij}:

$$x_{ij} := \begin{cases} 1 & \text{if job } i \text{ is in } [j]\text{-th position;} \\ 0 & \text{otherwise.} \end{cases}$$

Then the $mSAL$ problem is formulated as:

$$(P) \quad MIN \qquad Z$$

$$s.t.$$

$$\sum_{j=1}^{m} x_{ij} = 1 \qquad i = 1, \ldots, n \tag{6.1}$$

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad j = 1, \ldots, n \tag{6.2}$$

$$x_{ij} \in \{0,1\} \tag{6.3}$$

$$\sum_{i=1}^{n} \sum_{j=1}^{k} M_{ij} x_{i,k+1-j} \leq Z \qquad k = 1, \ldots, m-1 \tag{6.4}$$

$$\sum_{i=1}^{n} \sum_{j=1}^{m} M_{i,m-j+1} x_{i,k+j-1} \leq Z \qquad k = 1, \ldots, n-m+1 \tag{6.5}$$

$$\sum_{i=1}^{n} \sum_{j=0}^{k} M_{i,m-j} x_{i,n+j-k} \leq Z \qquad k = 0, 1, \ldots, m-2 \tag{6.6}$$

Equations (1)-(3) correspond to the assignment constraints, and equations (4)-(6) correspond to production cycles $1, 2, \ldots, n+m-1$. In particular, equation (4) corresponds to production cycles $1, 2, \ldots, m-1$, (5) to production cycles $m, m+1, \ldots, n$ and (6) to production cycles $n+1, \ldots, n+m-1$. Let $\Lambda_1, \Lambda_2, \Lambda_3$ be the set of constraints described by (4), (5), (6) respectively and $\Lambda = \Lambda_1 \bigcup \Lambda_2 \bigcup \Lambda_3$. For a given vector of multipliers:

$$\mu = \{\mu_\lambda : \lambda \in \Lambda, 0 \leq \mu_\lambda \leq 1, \sum_{\lambda \in \Lambda} \mu_\lambda = 1\}$$

the multiplier μ_λ corresponds to the λ-th production cycle and the surrogate relaxation of (P) is given by:

$$\begin{aligned} S(\mu) = MIN \quad & Z \\ s.t. \quad & \\ & \textstyle\sum_{\lambda \in \Lambda_1} \mu_\lambda \sum_{i=1}^{n} \sum_{j=1}^{k} M_{ij} x_{i,k+1-j} + \\ & \textstyle\sum_{\lambda \in \Lambda_2} \mu_\lambda \sum_{i=1}^{n} \sum_{j=1}^{m} M_{i,m-j+1} x_{i,k+j-1} + \\ & \textstyle\sum_{\lambda \in \Lambda_3} \mu_\lambda \sum_{i=1}^{n} \sum_{j=0}^{k} M_{i,m-j} x_{i,n+j-k} \leq Z \qquad (6.7) \\ & (1), (2), (3). \end{aligned}$$

Thus, to solve $S(\mu)$ optimally, we can equivalently solve:

$$\begin{aligned} S(\mu) = \; & MIN \textstyle\sum_{\lambda \in \Lambda_1} \mu_\lambda \sum_{i=1}^{n} \sum_{j=1}^{k} M_{ij} x_{i,k+1-j} + \\ & \textstyle\sum_{\lambda \in \Lambda_2} \mu_\lambda \sum_{i=1}^{n} \sum_{j=1}^{m} M_{i,m-j+1} x_{i,k+j-1} + \\ & \textstyle\sum_{\lambda \in \Lambda_3} \mu_\lambda \sum_{i=1}^{n} \sum_{j=0}^{k} M_{i,m-j} x_{i,n+j-k} \qquad (6.8) \\ s.t. \quad & (1), (2), (3). \end{aligned}$$

This formulation corresponds to solving the assignment problem $AP(\mu)$:

$$\begin{aligned} AP(\mu) = \; & MIN \textstyle\sum_{ij} c_{ij} x_{ij} \\ s.t. \quad & (1), (2), (3). \end{aligned}$$

where

$$c_{ij} := \sum_{r=1}^{m} \mu_{j+r-1} M_{ir}.$$

Let $X(\mu)$ be the solution for $S(\mu)$. If $Z(X)$ is the value of (P) for the solution X and Z^* the optimal value for (P), then:

$$S(\mu) \leq Z^*, \quad and$$

$$Z(X) \geq Z^* = \min_X \{Z(X)\}.$$

Hence, the solution of $S(\mu)$ can be used to obtain lower bounds for (P). It is essential in this algorithm to determine the multipliers μ_λ that result to tight lower bounds. The following heuristic constructs multiplier vectors that have shown empirically to be very satisfactory. For given vector μ the assignment X found by the assignment problem (8), produces a feasible sequence for $mSAL$, say S. Starting with S we perform pairwise interchanges of jobs systematically, until no improving interchange is possible. This procedure is known in the literature as 2OPT heuristic, and has been successfully applied to difficult combinatorial problems including the Traveling Salesman problem in Lin and Kernighan (1973). The resulting sequences produce upper bounds for $mSAL$ and thus, $SR1$ produces both lower and upper bounds.

Heuristic SR1

Input : M_{ij} $i = 1, 2, \ldots, n$, $j = 1, 2, \ldots, m$

Output: Lower and Upper bounds (LB, UB resp.), on the optimal manpower level

[1] $UB := +\infty$

[2] $LB := 0$

[3] Let $\mu_k := \frac{1}{n+m-1}$, $k = 1, 2, \ldots, n+m-1$

[4] Solve the (AP) with $c_{ij} := \frac{1}{n+m-1} \sum_{k=1}^{m} M_{ik}$, $1 \leq i, j \leq n$

[5] Let S be the resulting sequence and Z_S the corresponding value.

[6] *Repeat*

[7] *If* $(Z_S > LB)$ *then* $LB := Z_S$

[8] Apply 2OPT to S. Let S' the new sequence.

[9] *If* $(M(S') < UB)$ *then* $UB := M(S')$

[10] Compute the per cycle manpower req's of S', say $R_1, R_2, \ldots, R_{n+m-1}$.

[11] Let $\mu_k := \frac{R_k}{\sum_{k=1}^{n+m-1} R_k}$, $k = 1, 2, \ldots, n+m-1$.

[12] Solve the (AP) with $c_{ij} := \sum_{r=1}^{m} \mu_{j+r-1} M_{ir}$, $1 \leq i, j \leq n$

[13] Let S be the resulting sequence and Z_S the corresponding value.

[14] *Until* $(|LB - Z_S| < 1)$

As seen, SR1 terminates when a new lower bound is found that is within 1 unit away from the incumbent lower bound. A slight modification of SR1 is to replace lines [3] and [11] by

[3]′ Let

$$\mu_k := \begin{cases} \frac{1}{n-m+1} & \text{for k=m,...,n;} \\ 0 & \text{otherwise.} \end{cases}$$

[11]′ Let

$$\mu_k := \begin{cases} \frac{R_k}{\sum_{k=m}^{n} R_k} & \text{for k=m,...,n;} \\ 0 & \text{otherwise.} \end{cases}$$

respectively. We refer to this modification as *SR2*. Generating multipliers by SR2, we reflect the intuition that the maximum manpower requirement is very likely to be attained during one of the production cycles $k = m, \ldots, n$ because during these cycles all stations are busy; in contrast to all other cycles that utilize $m-1$ or fewer stations. This observation is certainly valid when the M_{ij}'s appear to take on random values from a medium size interval $[l, u]$.

Having generated the heuristics SR1 and SR2 that produce both lower and upper bounds we turn our attention to discovering other fast heuristics which have the potential to perform better. The difficulty of $mSAL$ lies in the observation that once we determine the schedule of two consecutive stations $j, j+1$ this partial schedule

determines the schedule of all m stations. For example consider a 4-station assembly line and suppose that stations ST_1, ST_2 have being scheduled. If $M_{i,2}$ is assigned to ST_2 for period k, then $M_{i,3}$ must be assigned to ST_3 at period $k+1$ and $M_{i,4}$ must be assigned to ST_4 at period $k+2$. This fact indicates that a local decision (such as scheduling 2 consecutive stations) determines the manpower value of a heuristic. However, since a local decision does not account for all the stations this value is likely to be very bad. Consider for instance the following greedy heuristic which is of the same nature to the ones used successfully for scheduling multiple machines in Cheng and Sin (1990), and crew scheduling problems in Coffman and Yannakakis (1984), and in Hsu (1984), etc.

Heuristic H_1

Input : M_{ij} $i = 1, 2, \ldots, n$, $j = 1, 2, \ldots, m$

Output: Upper bound UB on the optimal manpower level

$UB := 0$

Place jobs in J in a list L at random

for $i := 1$ *to* n *do begin*

Assign the job j at the top of L to position $[i]$ that results to minimum increase to the current manpower requirements over any cycle;

in case that several such positions exist, choose the one with smallest index

Remove j from L

Let S be the resulting partial sequence

if $(M(S) > UB)$ *then* $UB := M(S)$

endfor

To illustrate H_1 we apply it on the problem given at Table 6.2. Let L be the list $\{J_1, J_2, J_3, J_4\}$. H_1 puts J_1 at position [1]. Then J_2 is put at position [2] because this

Table 6.2. An example for the insertion Heuristic H_1

Job	Manpower Req's		
	M_{i1}	M_{i2}	M_{i3}
J_1	5	2	3
J_2	1	4	7
J_3	6	8	4
J_4	2	1	3

is the smallest index position that minimizes the current maximum manpower level. J_3 is then scheduled at position [4] where the current manpower requirement equals 13 (while if J_3 was scheduled at position [3] the current manpower requirement would equal 15). Finally, J_4 goes to position [3]. The final schedule S looks like:

[1]	5	2	3				$\leftarrow$	J_1
[2]		1	4	7			$\leftarrow$	J_2
[3]			2	1	3		$\leftarrow$	J_4
[4]				6	8	4	$\leftarrow$	J_3

Note that $M(S) = 14$ and is attained during cycle 4. We will demonstrate however that H_1 can have very bad performance in the worst case.

Theorem 15 $M_{H_1} \leq mM^*$ *and the bound is tight.*

Proof. For a given number of stations m, consider the instance, parameterized by M, that contains m jobs with manpower requirements $(M, M, \ldots, M)$ and $(m-1)^2$ jobs with manpower requirements $(0, 0, \ldots, 0)$.

An optimal solution for this instance is the sequence which has a $(M, M, \ldots, M)$ job at positions $1, m+1, 2m+1, \ldots, m^2-m+1$ and a $(0, 0, \ldots, 0)$ job at all other positions. One can check that for this schedule every period requires M workers and hence $M^* = M$.

Suppose that the job set is placed top to bottom in L in the following order. The top $(m-1)^2$ elements of L are $(0, 0, \ldots, 0)$ jobs and the bottom m elements

are $(M, M, \ldots, M)$ jobs. When H_1 is applied on L, the top job goes to position 1, the next job to position 2, and so on. The resulting sequence is $(m-1)^2$ jobs with requirements $(0, 0, \ldots, 0)$, followed by the m jobs with requirements $(M, M, \ldots, M)$ and hence $M_H = mM = mM^*$ which proves that the bound mM^* is tight. □

Apart from these negative results we can use O-2SAL to reduce the bound mM^* to $(m-1)M^*$. The heuristic H_2 utilizes O-2SAL as a subroutine.

Heuristic H_2

Input : M_{ij} $i = 1, 2, \ldots, n$, $j = 1, 2, \ldots, m$

Output: Upper bound UB on the optimal manpower level

$UB := +\infty$

for $st := 1$ *to* $m-1$ *do begin*

Apply O-2SAL on stations st, $st+1$

Let S_{st} be the resulting sequence

if $(M(S_{st}) < UB)$ *then* $UB := M(S_{st})$ and save S_{st}

endfor

Suppose that the sequence S obtained by H_2 is $S = S_{st}$. Then the manpower requirement for ST_{st}, ST_{st+1} is at most M^*, by Theorem 12. An argument similar to Theorem 14 shows that the manpower required by all other stations (except ST_{st}, ST_{st+1}) does not exceed $(m-2)M^*$. Hence, $M(S) \leq M^* + (m-2)M^* = (m-1)M^*$. The next theorem summarizes this result which shows that the error bound has been improved by M^*.

Theorem 16 $M_{H_2} \leq (m-1)M^*$.

The following example shows that the error bound is tight for $m = 3$.

Example 4: Consider an instance, parameterized by M, consisting of n jobs $(0, M, 0)$,

n jobs $(0,0,M)$ and n jobs $(M,0,0)$. An optimal sequence for this instance is the one where positions $1,4,7,\ldots,3n-2$ are occupied by jobs $(0,M,0)$, positions $2,5,8,\ldots,3n-1$ are occupied by jobs $(0,0,M)$ and positions $3,6,9,\ldots,3n$ are occupied by jobs $(M,0,0)$. By means of this sequence we conclude that $M^* = M$.

The sequence S_1 obtained by H_2 is the one where the first n jobs are $(0,M,0)$, the next n jobs are $(M,0,0)$ and the last n jobs are $(0,0,M)$. It so happens that $S_2 = S_1$ for this particular instance. We observe that $M(S_1) = M(S_2) = M$ while the sequence $S = S_1 = S_2$ results to $M(S) = 2M = (m-1)M^*$. Hence the bound $(m-1)M^*$ is tight. □

Another heuristic based on O-2SAL, is a simulation of the CDS heuristic of Campbell *et al.* (1970) for the m-machine flowshop problem. We refer to this heuristic as H_3.

Heuristic H_3

Input : M_{ij} $i = 1,2,\ldots,n$, $j = 1,2,\ldots,m$

Output: Upper bound UB on the optimal manpower level

$UB := +\infty$

for k:=1 *to* $m-1$ *do begin*

Let $M'_{i1} := \sum_{j=1}^{k} M_{ij}$ $\quad i := 1,\ldots,n$

Let $M'_{i2} := \sum_{j=m-k+1}^{m} M_{ij}$ $\quad i := 1,\ldots,n$

Apply O-2SAL with respect to (M'_{i1}, M'_{i2}) $\quad i := 1,\ldots,n$

Let S be the resulting sequence

If $(M(S) < UB)$ *then* $UB := M(S)$

endfor

The heuristics H_2, H_3 utilize O-2SAL to produce a sequence which is optimal for two stations and then apply this sequence for the m-station problem. However, this

approach is myopic since it considers only two stations at a time. An alternative approach is to produce sequences that are suboptimal for two stations but have good performance for the m-station problem. To do this systematically, we do as follows: Given two consecutive stations of the assembly line, say $st1$, $st1+1$, with manpower requirements M_{r1}, M_{r2}, $r = 1, 2, \ldots, n$ respectively, we order the entries M_{r1}, M_{r2}, as in the bipartite graph B of Section 3. Let $1 \leq i < j \leq n+1$, be indices such that $M_{i+1,1} \geq M_{i2}$ and $M_{j+1,1} \leq M_{j2}$ in this ordering. Then, the multiplier μ_{ij} indicates the value that equalizes the difference $(M_{i2} - M_{j2})$ and $(M_{j+1,1} - M_{i+1,1})$ of manpower levels, i.e. $\mu_{ij}(M_{i2} - M_{j2}) = (1 - \mu_{ij})(M_{j+1,1} - M_{i+1,1})$ and is given by:

$$\mu_{ij} := \frac{M_{j+1,1} - M_{i+1,1}}{(M_{i2} - M_{i+1,1}) - (M_{j2} - M_{j+1,1})}.$$

If $0 \leq \mu_{ij} \leq 1$, we multiply the entries M_{r1} by μ_{ij} and all entries M_{r2} by $1 - \mu_{ij}$, for $r = 1, 2, \ldots, n$. Then we apply O-2SAL for this modified 2-station problem and the resulting sequence is evaluated for the m-station problem.

Intuitively, this approach manipulates the values $M_{i2}, \ldots, M_{j2}$ and $M_{i+1,1}, \ldots, M_{j+1,2}$ so that the assignments made at step 1 of O-2SAL when applied for the modified problem, are different to those made when applied to the original problem. Since the multiplier μ_{ij} equalizes the differences $(M_{i2} - M_{j2})$ and $(M_{j+1,1} - M_{i+1,1})$, the optimal sequence for the modified problem, say S_M, is very likely to be near optimal for the original 2-station problem. Our last heuristic H_4 exploits the possibility that a minor loss due to suboptimality of S_M for the 2-station problem, is offset by the gains of S_M when applied to the m-station problem. Indeed, as we show empirically in the next section, the performance of H_4 dominates all the heuristics developed so far. The following procedure outlines the heuristic.

Heuristic H_4

Input : M_{ij} $i = 1, 2, \ldots, n$, $j = 1, 2, \ldots, m$

Output: Upper bound UB on the optimal manpower level

$UB := +\infty$

for $st1 := 1$ *to* $m-1$ *do begin*

for any two production cycles k, l, $1 \leq k < l \leq n-1$, *do begin*

if $M_{k,st1+1} \leq M_{k+1,st1}$ and $M_{l,st1+1} \geq M_{l+1,st1}$ *then begin*

compute $\mu_{kl} = \frac{M_{l+1,st1} - M_{k+1,st1}}{(M_{k,st1+1} - M_{k+1,st1}) - (M_{l,st1+1} - M_{l+1,st1})}$

if $0 \leq \mu_{kl} \leq 1$ *then begin*

Apply O-2SAL with respect to $M'_{i1} = \mu_{kl} M_{i,st1}$, $M'_{i2} = (1 - \mu_{kl}) M_{i,st1+1}$

Let S_{kl} be the resulting sequence

if $M(S_{kl}) < UB$ then set $UB := M(S_{kl})$ and save S_{kl}

endif

endif

endfor

endfor

In the next section we perform computational experiments with all the heuristics developed in this section. It will be seen that although the worst case behavior of H_1, H_2, H_3, H_4 may be bad for assembly lines with many stations, some of these heuristics have very good average performance.

6.6 Computational Results

We test the various heuristics on random problems of size $m = 3, 6, 9, 12$ and $n = 10, 15, 20$. For each combination (m, n), the entries M_{ij} assume random values taken from the discrete uniform distribution on the interval $[10, 20]$, which is a realistic range for many applications. For each combination, we run 10 problems on a PC compatible running at 66 MHz.

We report the performance of H_1, H_2, H_3, H_4 when $2OPT$ is applied to all the sequences produced by the heuristics. In all cases, the performance of the best

Table 6.3. Relative gap of $SR1, SR2, H_1, H_2, H_3, H_4$ when $M_{ij} \in [10, 20]$.

Relative Gap = $\frac{UB_H - LB}{LB}$ 100% where $LB = \max\{LB_{SR1}, LB_{SR2}\}$													
m	n	$SR1$		$SR2$		H_1		H_2		H_3		H_4	
		Mean	Max	Mean	Max	Mean	Max	Mean	Max	Mean	Max	Mean	Max
3	10	15.3	23.5	12.7	17.2	8.2	10.0	7.2	10.0	9.7	12.5	6.7	10.2
	15	18.3	31.3	12.0	14.7	14.8	19.0	10.6	13.9	9.2	11.9	6.9	9.5
	20	16.5	25.8	14.4	21.9	13.1	17.7	10.4	11.9	11.3	15.5	6.8	9.5
6	10	17.1	21.8	10.8	17.5	7.9	11.2	5.9	8.7	7.1	15.0	4.4	6.2
	15	13.3	15.7	7.5	8.8	8.8	12.2	7.9	12.2	7.9	10.9	6.4	8.3
	20	16.7	23.1	11.5	17.7	10.4	14.6	8.0	9.6	8.7	11.4	6.6	7.3
9	10	30.6	53.0	15.3	37.9	10.0	12.9	6.9	10.1	8.4	11.1	6.4	9.2
	15	18.7	20.9	8.6	10.2	9.2	13.4	7.0	7.5	9.5	13.1	4.7	5.8
	20	17.0	24.0	11.6	22.3	8.5	10.3	7.9	9.0	7.4	9.8	5.9	6.3
12	10	26.1	40.4	9.5	23.0	8.1	10.2	6.3	7.7	6.1	12.6	4.5	5.9
	15	24.9	34.7	11.0	19.0	8.2	11.4	6.7	7.4	6.6	9.4	4.9	6.0
	20	20.2	25.7	10.3	15.6	7.8	10.1	6.9	8.5	7.2	13.2	5.2	6.1

generated sequence is reported. Recall that H_1 produces only 1 sequence, H_2, H_3 produce $m-1$ sequences each, and H_4 produces an unpredictable number of sequences. Also, we report the performance of the heuristics $SR1$ and $SR2$. Since both of these heuristics produce lower and upper bounds, the performance of $SR1$ is measured by the gap $\frac{UB_{SR1} - LB_{SR1}}{LB_{SR1}}$ where UB_{SR1} is the lowest upper bound of all sequences produced by $SR1$ and LB_{SR1} is the greatest lower bound of all iterations of $SR1$. Similar results are reported for $SR2$. All other heuristics were compared against $LB := \max\{LB_{SR1}, LB_{SR2}\}$. The mean and maximum (over 10 test problems) relative gap $\frac{UB_H - LB}{LB}$ is reported for the heuristics H_1, H_2, H_3, H_4. These results are presented in Table 6.3 and we summarize them next.

1) *SR2 produces tight lower bounds.*

Both SR1, SR2, converge in 2 or 3 iterations to a lower bound that is within 1 unit from the incumbent lower bound, and therefore their computational requirements are minor (see Table 6.4). SR2 produces better lower bounds for our test generator, but SR1 tends to produce better upper bounds than SR2. However, none of these two heuristics produces good upper bounds. Specifically, the relative gap ranges in 15-50% for SR1 and 8-38% for SR2.

2) H_4 *outperforms all other heuristics.*

The heuristic H_1 has a relative gap 7-15%. Recall that H_1 produces only one sequence, and as expected has negligible requirements in computing time (see Table 6.4). Now we compare H_2 with H_3 since both produce $m-1$ sequences. H_2 gives slightly better solutions than H_3, ranging 6-11%. The time requirements of H_2, H_3, never exceed 9 seconds of execution time, which adds to their appeal. We conclude that among the heuristics H_1, H_2, H_3 that have negligible time requirements, H_2 has the best average performance. However, none of these two heuristics is robust, in the sense that the maximum relative gap can be up to 14% for H_2 and 15.5% for H_3.

The clear winner however, of all proposed algorithms, is H_4. The relative gap for this heuristic ranges in 4-7% while the maximum relative gap is no more than 10.2%. This means that the heuristic is robust and therefore very reliable in giving near optimal solutions.

3) *All heuristics have negligible to minor time requirements.*

As we see in Table 6.4, H_1, H_2 and H_3 have negligible time requirements. The heuristic H_4 however, generates many more sequences than the others, but still only a tiny portion of the $n!$ possible sequences. For $n=20$ and $m=12$, which is considered to be a large problem size, H_4 generates only 760 sequences on the average, and requires about 5 minutes of PC execution time.

4) *Problem difficulty decreases with the ratio* $\frac{\max_{ij} M_{ij}}{\min_{ij} M_{ij}}$.

In Section 6.4 we showed that for every sequence S we have that

$$\frac{M(S)}{M^*} \leq \frac{\max_{ij} M_{ij}}{\min_{ij} M_{ij}}.$$

This is an indication that problem difficulty is proportional to the ratio $\frac{\max_{ij} M_{ij}}{\min_{ij} M_{ij}}$. To exploit the validity of this assertion we compute in Table 6.5 the average relative gap for all heuristics when M_{ij} takes on values uniformly from the interval $[20, 35]$. In this case $\frac{\max_{ij} M_{ij}}{\min_{ij} M_{ij}} = 1.75$ versus a value of 2.0 in Table 6.3 where $M_{ij} \in [10, 20]$. Although

Table 6.4. Time requirements of Heuristics, in seconds.

m	n	$SR1$	$SR2$	H_1	H_2	H_3	# seq's	H_4
	10	0.05	0.06	0.02	0.00	0.04	28.0	0.34
3	15	0.12	0.18	0.05	0.08	0.07	71.0	2.66
	20	0.27	0.44	0.11	0.17	0.20	129.2	13.18
	10	0.06	0.11	0.02	0.14	0.12	73.0	1.85
6	15	0.20	0.32	0.08	0.42	0.38	173.0	15.53
	20	0.47	0.70	0.23	0.90	0.82	336.2	65.87
	10	0.06	0.12	0.01	0.26	0.23	113.0	3.80
9	15	0.22	0.37	0.13	0.87	0.85	294.8	36.16
	20	0.54	0.87	0.30	2.46	1.77	539.0	165.55
	10	0.18	0.06	0.03	0.55	0.23	162.7	7.23
12	15	0.37	0.54	0.19	1.87	1.46	398.0	65.74
	20	0.85	1.55	0.44	3.52	3.13	760.0	321.46

the difference of the two ratios is small, the relative gaps reported in Table 6.5 are significantly smaller than the corresponding gaps of Table 6.3. This observation shows that it is easier to discover good heuristic solutions when the variability of the M_{ij} values is small.

To better evaluate our heuristics we compare the lower and upper bounds developed so far with the optimal solution to formulation (P) presented in Section 6.5. For each (n, m) combination presented in Table 6.6 we generated 10 problems with M_{ij} uniformly distributed in $[10, 20]$. To solve these problems we used the AMPL interface (see Fourer *et al.*, 1993) and the CPLEX solver. We performed the experiment on a PC compatible running at 66 MHz. The execution time allowed for each of the problems tested was 1 hour. Problem sizes that were larger than those reported in Table 6.6 required significantly more time thus making their testing prohibitive. We report the average deviation of the lower bound LB (i.e. the maximum of the lower bounds obtained by SR1, SR2) from the optimal solution as well as the average deviation of the heuristic solutions obtained by H_1, H_2, H_3, H_4 from the optimal.

Table 6.5. Relative gap of $SR1, SR2, H_1, H_2, H_3, H_4$ when $M_{ij} \in [20, 35]$.

Relative Gap = $\frac{UB_H - LB}{LB}$100% where $LB = \max\{LB_{SR1}, LB_{SR2}\}$													
m	n	$SR1$		$SR2$		H_1		H_2		H_3		H_4	
		Mean	Max	Mean	Max	Mean	Max	Mean	Max	Mean	Max	Mean	Max
3	10	6.8	9.7	11.4	15.6	8.1	14.0	5.1	7.2	5.6	10.1	4.4	5.2
	15	9.7	13.1	12.3	18.3	10.3	15.2	6.8	10.5	7.2	10.1	5.4	8.7
	20	8.3	10.4	11.2	13.9	8.6	11.6	6.7	8.9	8.1	10.1	6.0	8.9
6	10	5.6	18.2	16.4	30.0	6.8	10.1	4.0	5.6	3.8	5.50	2.9	5.4
	15	6.7	8.0	12.4	14.6	7.4	11.4	5.3	6.2	4.9	7.2	4.5	5.4
	20	8.1	15.3	12.2	21.8	8.1	13.2	5.8	7.0	6.3	8.7	4.8	6.1
9	10	7.8	16.7	29.8	43.7	3.8	6.9	2.6	4.4	3.1	5.0	2.1	3.2
	15	7.2	13.3	17.7	19.4	5.6	8.2	4.2	4.7	4.3	4.7	3.1	3.5
	20	9.0	15.4	15.8	20.5	6.8	8.3	5.4	5.8	6.1	7.7	4.5	4.7
12	10	8.4	13.2	30.1	41.2	3.1	5.2	2.1	4.7	2.8	4.4	1.9	2.8
	15	7.5	14.8	27.0	35.2	5.7	9.3	3.9	5.2	3.8	5.0	3.2	3.9
	20	5.7	6.6	17.8	19.0	6.5	8.0	4.5	4.9	4.2	4.9	3.5	3.9

Table 6.6. Relative deviation of LB, H_1, H_2, H_3, H_4 from optimal solution.

		$\frac{LB-M^*}{M^*}$100%	$\frac{UB_H-M^*}{M^*}$100%			
m	n	LB:=max$\{LB_{SR1}, LB_{SR2}\}$	H_1	H_2	H_3	H_4
3	6	3.28	1.86	0.97	2.29	0.93
	9	1.76	7.55	4.02	3.46	1.76
	12	4.32	6.12	3.32	5.53	0.59
6	6	4.34	4.02	2.84	2.65	1.20
	9	3.57	5.96	1.21	1.82	0.60
	12	5.28	2.87	1.14	4.59	0.57
9	6	3.03	2.83	1.12	1.33	0.49
	9	2.91	5.02	0.93	1.00	0.51
	12	4.35	4.09	1.09	0.82	0.54
Average		3.64	4.48	1.84	2.61	0.79

Table 6.6 suggests that the relative gaps reported in Tables 6.3 and 6.5 are mostly attributed to the deviation of LB from the optimal solution M^*. This is very reasonable if we recall that in our surrogate relaxation approach $mSAL$ is reduced to an assignment problem which does not take into account the structure of the problem; it only considers the values M_{ij}. Since the relative gaps reported in Tables 6.3 and 6.5 are relatively small and largely attributed to the deviation of LB from M^*, some of the heuristic solutions are near optimal. This is also demonstrated in Table 6.6 where the heuristics H_2, H_3, H_4 deviate from optimality an average of 1.84%, 2.61%, 0.79% respectively. The success of the heuristics is intuitively justified from the fact that the sequences produced by each of H_2, H_3, H_4 are selected so that a pair of adjacent stations is scheduled optimally (or near optimally in the case of H_4).

In light of the near optimal performance of the heuristics H_2, H_3, H_4 and their negligible computational requirements, we feel confident to suggest our heuristics as an efficient and effective tool for solving the $mSAL$ problem, especially for large size problems. In the next section we consider multicriteria problems on assembly lines.

6.7 Extensions

Having developed several good heuristics for $mSAL$ we can explore the trade-off between a trial value of c and the corresponding manpower level M_c^*. This is especially useful in the design of a transfer line where we would like to see graphically several (c, M_c^*) points before we make decisions. The designer of an assembly line often wants to reduce the production cycle by increasing the number of workers in the line. Usually the number of additional workers is estimated in proportion to the desired improvement of cycle time. For example, if the cycle time is 10 and one wants to reduce it to 5, the original level of $M^* = 100$ is doubled to 200. This approach, although reasonable, might use many more workers than needed in which case the management absorbs unnecessary costs. As we will show, a more accurate approach

is to implement any of the heuristics that we developed and provide the manager with the trade-off curve of the length of the cycle c and the corresponding manpower level M_c^*. We illustrate this idea for the case $m = 2$, using O-2SAL. Similar results can be obtained for $m > 2$, by applying any of the heuristics presented in the previous section, in place of O-2SAL.

Consider a 3-job example with processing time requirements $(54, 35), (52, 63)$, $(45, 58)$ where the first component of each pair represents the total man-hours required to complete the subassembly in the first station. Similarly for the second component of each pair. We want to find M_c^* for the values $c = 6, 7, 8, 9, 10$. Although the number of workers able to work in some station may be limited by design specific factors, let us assume for the sake of exposition that $M_{ij} = \lceil \frac{p_{ij}}{c} \rceil$ where p_{ij} denotes the man-hours required by job J_i at station ST_j. Then, the manpower requirements of the 3 jobs for $c = 10$ are $(6, 4), (6, 7), (5, 6)$. Similarly we can find the manpower requirements for all jobs. The solution obtained by O-2SAL for this instance is $6 - 7 \cdots 5 - 6 \cdots 6 - 4$ and $M_{10}^* = 12$. The manpower level for the various values of c under the assumption that $M_{ij} = \lceil \frac{p_{ij}}{c} \rceil$ are given in Table 6.7. From these results, one can see that a workforce

Table 6.7. Cycle length versus manpower level

c	6	7	8	9	10
M_c^*	19	16	14	12	12

of 12 workers is able to process the jobs in a cycle of 9 periods and therefore the cycle $c = 10$ is superfluous. This means that when $c = 10$, although 12 workers are necessary in a particular cycle, some of them stay idle during other cycles. The same 12 workers can execute the 3 jobs in 9 periods in which case some of the workers' idle time is eliminated. Also we can see that an increase of the workforce by 7 units can reduce the production cycle time from 10 units to 6.

An implicit assumption in this chapter is that all workers are trained to work in all stations. When the level of specialization is low, as is usually the case for assembly operations, this assumption is valid. In case that there are many levels of worker specialization, Ebeling and Lee (1993) have discussed the effectiveness of cross-training. Also, Yano and Rachamadugu (1991) discuss the practice of alleviating work overload by dispatching *utility workers* to the point of the overload. Utility workers are employees with a broad base of cross-training who are capable of assisting on multiple tasks along the assembly line. Although the problem of crosstraining can be very difficult depending on the relevant costs, we can use the algorithms presented for the $mSAL$ model to find an upper bound on the number of utility workers.

Consider an instance of $mSAL$ with requirements $(M_{i1}, M_{i2}, \ldots, M_{im})$ for $i := 1, 2, \ldots, n$. Then, in every production cycle, station j requires at least $A_j := \min_i M_{ij}$ workers. If we specialize A_j $j : 1, 2, \ldots, m$ workers for station ST_j only, it remains to find the minimum number of utility workers (i.e. workers that are trained for all stations) for the reduced problem that has manpower requirements $(M_{i1} - A_1, M_{i2} - A_2, \ldots, M_{im} - A_m)$ for $i := 1, 2, \ldots, n$. By construction, an optimal sequence for the reduced problem, induces an optimal sequence for the original problem (where all workers are able to work in all stations) and vice versa.

This approach divides the workforce into two categories; namely workers that are trained for one station only and workers that are trained for all stations. This policy may be realistic in applications where the requirements of the different jobs are about the same for each station.

6.8 Conclusion

In this research we addressed the manpower planning problem in transfer lines. Our problem is inspired by a real problem in industry and we indicated several other sectors that share similar concerns. We developed an optimal algorithm for the case of

two stations (cells) and several heuristics that produce near optimal solutions for the general problem. We showed how these tools can be used by a manager to determine his production cycle for a given workforce level, or the number of specialized and utility workers. In our future research we will include cross-training constraints to our model, to address the group of manufacturers where job specialization or machine flexibility is limited.

CHAPTER 7

FUTURE RESEARCH

There is a sentiment on the part of managers that the design of a manufacturing process is based on each individual's experience. As a result there is not much communication between managers and researchers regarding process design. The main reason is that the trade-offs between flexibility and performance of flexible designs has not received significant attention prior to this thesis. In this thesis we developed a framework according to which we compare various designs on a scientific basis.

We use our framework to come up with management technology as well as managerial guidelines that will allow us to advance process choice decisions to a science rather than an empirical way of making decisions. We studied several flowshop environments. The basic characteristic according to which we categorized the various designs, is the mode of flexibility featured by each design. We considered processing as well as routing flexibility.

Other modes and production environments can be identified and studied in detail. The first environment to come in mind is that of a jobshop where different machines may possess different characteristics. Our framework can also be used on assembly lines as we demonstrate in this thesis where we compare the mixed model transfer line with the mixed model assembly line.

For every study undertaken in the field of process choice decisions the ultimate objective is twofold. First to develop better algorithms than the existing ones for managing the underline production systems and second to categorize the production environments that favor one design over another. We are particularly interested in

developing rules of thumb such as: "two parallel identical machines should be used instead of a flexible flowshop, unless 2% improvement in the throughput rate outperforms the additional equipment and scheduling costs attributed to the flexible flowshop". As we saw in Chapter 4, the two parallel identical machine environment (2P) is only 2% inferior to a flexible flowshop (2FFS). However, the 2P enjoys a simpler routing control structure and scheduling compared to that of 2FFS.

As it became apparent in Chapter 6, such studies may reveal exciting new conclusions as the concept of design for schedulability and the correspondence of the hardware and software aspects of computer systems with the flexibility and schedulability aspects of production systems. In addition, our efforts for identifying the important characteristics that distinguish one design from another, will help us to design new hybrid systems that better serve our production needs. This is the case in the mixed model transfer line where the issue of manpower control is directly addressed, thus leading to an assembly line that controls workforce tightly. More assembly systems can be studied starting with an assembly line appropriate for models of high variability; in that case the various stations could have different production cycles.

It is our wish that the research community will recognize the need for managerial guidelines regarding process choice decisions. Only then flexible manufacturing will assume a scientific rather than empirical stature.

REFERENCES

[1] Adam E. and Ebert R.J. (1992). Production and Operations Management. Fifth Edition. Prentice Hall, Englewood Cliffs, NJ.

[2] Afentakis P. (1986). Maximum Throughput in Flexible Manufacturing Systems. *Proceedings of the Second ORSA/TIMS Conference on Flexible Manufacturing Systems: Operations Research Models and Applications*, Elsevier Science Publishers B.V., Amsterdam, 509-521.

[3] Askin R.G. and Standridge C.R. (1993). Modeling and Analysis of Manufacturing Systems. Wiley, New York.

[4] Baker K.R. (1974). Introduction to Sequencing and Scheduling. Wiley, New York.

[5] Bard J.F. (1989). Assembly Line Balancing with Parallel Workstations and Dead Time. *International Journal of Production Research* 27(6):1005-1018.

[6] Bard J.F., Dar-El E. and Shtub A. (1992). An Analytic Framework for Sequencing Mixed Model Assembly Lines. *International Journal of Production Research* 30(1):35-48.

[7] Bartholdi J.J. III (1992). Balancing 2-Sided Assembly Lines (A Case Study). Working Paper.

[8] Bartholdi J.J. III and Eisenstein D.D. (1993). A Production Line That Balances Itself. Working paper.

[9] Baybars I. (1986). A Survey of Exact Algorithms for the Simple Assembly Line Balancing Problem. *Management Science* 32:909-932.

[10] Baybars I. and Frieze A. (1986). Expected Behavior of Line Balancing Heuristics. *IMA Journal of Mathematical Management* 10:304-335.

[11] Buten R.E. and Shen V.Y. (1973). A Scheduling Model for Computer Systems with Two Classes of Processors. *Sagamore Computer Conference on Parallel Processing* 130-138. Princeton University.

[12] Campbell H.G., Dudek R.A. and Smith M.L (1970). A Heuristic Algorithm for the n Job, m Machine Sequencing Problem. *Management Science* 16:630-637.

[13] Carlier J. (1987). Scheduling Jobs with Release Dates and Tails on Identical Machines to Minimize the Makespan. *European Journal of Operations Research* 29:298-306.

[14] Chen F.F. and Adam E.E. Jr. (1991). The Impact of Flexible Manufacturing Systems on Productivity and Quality. *IEEE Transactions on Engineering Management* 38:33-45.

[15] Cheng T.C.E. and Sin C.C.S. (1990). A State-of-the-art Review of Parallel-machine Scheduling Research. *European Journal of Operational Research* 47:271-292.

[16] Coffman E.G. and Yannakakis M. (1984). Permuting Elements Within Columns of a Matrix in Order to Minimize Maximum Row Sum. *Mathematics of Operations Research* 9:384-390.

[17] Dar-El E.M. (1977). Mixed Model Assembly Line Sequencing Problems. *Omega* 6(4):313-323.

[18] Dar-El E.M. and Cother R.F. (1975). Assembly Line Sequencing for Model-Mix. *International Journal of Production Research* 13(5):463-477.

[19] Ebeling A.C. and Lee C.-Y. (1993). Cross-training effectiveness and profitability. To appear in *International Journal of Production Research.*

[20] Erschler J., Roubellat F. and Thuriot C. (1985). Steady State Scheduling of a Flexible Manufacturing System with Periodic Releasing and Flow Time Constraints. *Annals of Operations Research* 3:333-353.

[21] Flynn B.B. and Jacobs R.F. (1987). An Experimental Comparison of Cellular (Group Technology) Layout with Process Layout. *Decision Sciences* 18:562-581.

[22] Fourer R., Gay D.M. and Kernighan B.W. (1993). AMPL: A Modeling Language for Mathematical Programming. The Scientific Press.

[23] French S. (1982). Sequencing and Scheduling: An introduction to the Mathematics of the Job Shop. Horwood, Chichester.

[24] Fry T.D., Wilson M.G. and Breen M. (1987). A Successful Implementation of Group Technology and Cell Manufacturing. *Production and Inventory Management* 28:4-6.

[25] Garey M.R. and Johnson D.S. (1979). Computers and Intractability. W.H. Freeman, San Francisco, CA.

[26] Ghosh, S. and C. Gaimon (1992). Routing Flexibility and Production Scheduling in a Flexible Manufacturing environment. *European Journal of Operational Research*, 60:344-364.

[27] Glover F. (1975) Surrogate Constraint Duality in Mathematical Programming. *Operations Research* 23:434-453.

[28] Gonzalez T. and S. Sahni (1978). Flowshop and Jobshop Schedules; Complexity and Approximation. *Operations Research* 26:36-52.

[29] Goodrich T.H. (1988). Just-in-Time with an Emphasis on Group Technology. *Manufacturing Systems* 6:78-79.

[30] Graham R.L. (1966). Bounds For Certain Multiprocessing Anomalies. *Bell System technical Journal* 45:1563-1581.

[31] Graham R.L. (1969). Bounds on Multiprocessing Timing Anomalies. *SIAM Journal of Applied Mathematics* 17:416-429.

[32] Gupta J.N.D. (1988). Two Stage Hybrid Flowshop Scheduling Problem. *Journal of the Operational Research Society* 39(4):359-364.

[33] Gupta J.N.D. and Tunc E.A. (1991). Schedules For a Two Stage Hybrid Flowshop With Parallel Machines at The Second Stage. *International Journal of Production Research* 29(7):1489-1502.

[34] Herrmann J.W. and Lee C.-Y. (1992). Three-Machine Look-Ahead Scheduling Problems. *Research Report* 92-23. Department of Industrial and Systems Engineering. University of Florida.

[35] Hsu W.-L. (1984). Approximation Algorithms for the Assembly Line Crew Scheduling Problem. *Mathematics of Operations Research* 9:376-383.

[36] Hyer N. (1984). The Potential of Group Technology for U.S. Manufacturing. *Journal of Operations Management* 4:183-202.

[37] Jackson J.R. (1955). Scheduling a Production Line to Minimize Maximum Tardiness. Research 43, Management Science Research Project, University of California, Los Angeles.

[38] Johnson S.M. (1954). Optimal Two and Three-Stage Production Schedules with Setup Times Included. *Naval Research Logistics Quarterly* 1:61-68.

[39] Kouvelis, P. and G. Vairaktarakis (1993). A Class of Two Stage Flexible Flowshops and Comparison of Their Throughput Performance. Submitted to *Management Science*.

[40] Langston M.A. (1978). Interstage Transportation Planning in The Deterministic Flow-Shop Environment. *Operations Research* 35(4):556-564.

[41] Lee C.-Y., Cheng T.C.E. and Lin B.M.T. (1993). Minimizing the Makespan in the 3-Machine Assembly-Type Flowshop Scheduling Problem. *Management Science* 39(5):616-625.

[42] Lee C.-Y. and Herrmann J.W. (1993). A Three-Machine Scheduling Problem with Look-Behind Characteristics. *Research Report* 93-11. Department of Industrial and Systems Engineering. University of Florida.

[43] Lee C.-Y. and Vairaktarakis G.L. (1993). Design For Schedulability: Look-behind And Look-ahead Flowshops. *Research Report* 93-23. Department of Industrial and Systems Engineering. University of Florida.

[44] Lin S. and Kernighan B.W. (1973). An Effective Heuristic Algorithm for the Traveling Salesman Problem. *Operations Research* 21:498-516.

[45] Luggen W.L. (1991). Flexible Manufacturing Systems. Prentice Hall, Englewood Cliffs, NJ.

[46] Maleki R.A. (1991). Flexible Manufacturing Systems: The Technology and Management. Prentice Hall, Englewood Cliffs, NJ.

[47] Miltenburg J. (1989). Level Schedules for Mixed-model Assembly Lines in Just-in-Time Production Systems. *Management Science* 35(2):192-207.

[48] Morton T.E. (1992). Heuristic Scheduling Systems. Graduate School of Industrial Administration, Carnegie Mellon University. Wiley, New York. In Press, 1992.

[49] Narasimhan S.L. and S.S. Panwalker (1984). Scheduling in a Two-Stage Manufacturing Process. *International Journal of Production Research* 22:555-564.

[50] Okamura K. and Yamashima H. (1979). A Heuristic Algorithm for the Assembly Line Model-Mix Sequencing Problem to Minimize the Risk of Stopping the Conveyor. *International Journal of Production Research* 17(3):233-247.

[51] Papadimitriou C.H. and Steiglitz K. (1982). Combinatorial Optimization: Algorithms and Complexity. Prentice Hall, Englewood Cliffs, NJ.

[52] Paul R.J. (1979). A Production Scheduling Problem in the Glass-Container Industry. *Operations Research* 22:290-302.

[53] Pinto P.A., Dannenbring D.G. and Khumawala B.M. (1981). Branch and Bound and Heuristic Procedures for Assembly Line Balancing With Parallel Stations. *International Journal of Production Research* 19:565-576.

[54] Rao T.B.K. (1970). Sequencing in the Order A, B with Multiplicity of Machines for a Single Operation. *Opsearch* 7:135-144.

[55] Shanker K. and Tzen Y.T. (1985). A Loading and Dispatching Problem in a Random Flexible Manufacturing System. *International Journal of Production Research* 23(3):579-595.

[56] Shriskandarajah C. and Ladet P. (1986). Some No-wait Shops Scheduling Problems. *European Journal of Operational Research* 24:424-445.

[57] Shriskandarajah C. and Sethi S.P. (1989). Scheduling Algorithms for Flexible Flowshops: Worst and Average Case Performance. *European Journal of Operational Research* 43:143-160.

[58] Stecke K.E. (1985). Design, Planning, Scheduling and Control Problems of Flexible Manufacturing Systems. *Annals of Operations Research* 3:3-12.

[59] Stecke K.E. (1992). Planning and Scheduling Approaches to Operate a Particular FMS. *European Journal of Operational Research* 61:273-291.

[60] Steiner G. and Yeomans S. (1993). Level Schedules for Mixed-model Just-in-Time Processes. *Management Science* 39(6):728-735.

[61] Thomopoulos N.T. (1967). Line Balancing sequencing for Mixed Model Assembly. *Management Science* 14(2):59-75.

[62] Thomopoulos N.T. (1970). Mixed Model Line Balancing With Smooth Station Assignments. *Management Science* 16(9):593-603.

[63] Wittrock R.J. (1985). Scheduling Algorithms for Flexible Flow Lines. *IBM J. Res. Develop.* 29:401-412.

[64] Wittrock R.J. (1988). An Adaptable Scheduling Algorithm for Flexible Flow Lines. *Operations Research* 26:445-453.

[65] Yano C. and Rachamadugu R. (1991). Sequencing to Minimize Work Overload in Assembly Lines with Product Options. *Management Science* 37(5):1-15.

BIOGRAPHICAL SKETCH

George Vairaktarakis was born in Athens, Greece, on July 10, 1964. He received his undergraduate degree in mathematics from the University of Patras, Greece, in July 1988. In August 1988 he came to the United States and joined the University of South Carolina at Columbia, South Carolina, where he received his Master of Arts degree in operations research. In August 1990 he joined the Department of Industrial and Systems Engineering of the University of Florida, Gainesville, Florida, where he pursued the Ph.D. degree in industrial engineering. Throughout his graduate studies he has been working as a graduate teaching and research assistant.

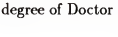
I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Chung-Yee Lee, Chairman
Associate Professor of Industrial
and Systems Engineering

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

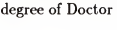
D. Jack Elzinga
Professor of Industrial
and Systems Engineering

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.



Suleyman Tufekci
Associate Professor of Industrial
and Systems Engineering

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.



Sartaj Sahni
Professor of Computer and Information
Sciences

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Sâhin S. Erengüç
Professor of Decision and Information Science

This dissertation was submitted to the Graduate Faculty of the College of Engineering and to the Graduate School and was accepted as partial fulfillment of the requirements for the degree of Doctor of Philosophy.

August 1994

Winfred M. Phillips
Dean, College of Engineering

Karen A. Holbrook
Dean, Graduate School